MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

TÍTULO COMPLETO DEL TFM

NOMBRE DEL AUTOR

20XX

MÁSTER uNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

trabajo fin de MÁSTER

**Título:** ……………….

**Autor:** D…………………….

**Tutor:** D. ……………….

**Ponente:** D. ………………

**Departamento:** …………

Miembros del Tribunal

**Presidente:** D. ……………

**Vocal:** D. …………..

**Secretario:** D. …………..

**Suplente:** D. ……………..

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: ………

Madrid, a de de 20…

**UNIVERSIDAD POLITÉCNICA DE MADRID**

**ESCUELA TÉCNICA SUPERIOR**

**DE INGENIEROS DE TELECOMUNICACIÓN**

****

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

TÍTULO COMPLETO DEL TFM

NOMBRE DEL AUTOR

20XX

summary

Maximum number of words: 500…

This paper studies mobile edge computing (MEC) networks where multiple wireless devices (WDs) choose to offload their computation tasks to an edge server. To conserve energy and maintain quality of service for WDs, the optimization of joint offloading decision and bandwidth allocation is formulated as a mixed integer programming problem. However, the problem is computationally limited by the curse of dimensionality, which cannot be solved by general optimization tools in an effective and efficient way, especially for large-scale WDs. In this paper, we propose a distributed deep learning-based offloading (DDLO) algorithm for MEC networks, where multiple parallel DNNs are used to generate offloading decisions.

We adopt a shared replay memory to store newly generated offloading decisions which are further to train and improve all DNNs. Extensive numerical results show that the proposed DDLO algorithm can generate near-optimal offloading decisions in less than one second.

KEYWORDS

Mobile edge computing, Offloading, Resource Allocation, Deep learning, LSTM

índice del contenido

[1. introducción y objetivos 6](#_Toc515605508)

[1.1. Introducción 6](#_Toc515605509)

[1.2. Objetivos 6](#_Toc515605510)

[2. desarrollo 2](#_Toc515605511)

[3. resultados 3](#_Toc515605512)

[4. conclusiones y líneas futuras 4](#_Toc515605513)

[4.1. Conclusiones 4](#_Toc515605514)

[4.2. Líneas futuras 4](#_Toc515605515)

[5. bibliografía 5](#_Toc515605516)

[anexo a: aspectos ÉTICOS, económicos, sociales y ambientales 6](#_Toc515605517)

[A.1 iNTRODUCCIÓN 6](#_Toc515605518)

[A.2 DESCRIPCIÓN DE IMPACTOS RELEVANTES RELACIONADOS CON EL PROYECTO 6](#_Toc515605519)

[A.3 ANÁLISIS DETALLADO DE ALGUNO DE LOS PRINCIPALES IMPACTOS 6](#_Toc515605520)

[A.4 CONCLUSIONES 6](#_Toc515605521)

[anexo b: presupuesto económico 7](#_Toc515605522)

# introduction and objectives

## INTRODUCTION

Over the last few decades Mobile Networks has evolved from technologies like 2G to 3G and from 3 to LTE (4G). This evolution is due to the increasing number of smart phones consuming data traffic during the last years. Traffic data like messaging, video streaming, P2P applications. Besides, this growth is expected to increase exponentially in the next years because of the process of of evolution of technologies such as mobile cloud computing and new mobile services. The impact of this large amount of traffic will produce extremely network load that with derive in an increase of the demand of the network bandwidth as data needs to be transmitted from the ME (Mobile User) to the cloud data centers and the way back.

According to a white paper from Cisco ‘Global Mobile Data Traffic Forecast Update, 2017–2022’

<https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html>

The number of estimated devices connected to the IoT will be around 50 billion by 2020. And mobile data traffic is predicted to continue doubling each year.

(Meter tabla del final de las conclusiones)

Global Mobile Data Traffic, 2017–2022

|  | **2017** | **2018** | **2019** | **2020** | **2021** | **2022** | **CAGR 2017–2022** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| By Application Category (PB per Month) | | | | | | | |
| Video | 6,821 | 12,051 | 19,279 | 29,149 | 42,734 | 60,889 | 55% |
| Non-Video | 4,691 | 6,959 | 9,281 | 11,621 | 14,064 | 16,604 | 29% |
| By Device Type (PB per Month) | | | | | | | |
| Smartphones | 10,132 | 17,172 | 26,122 | 37,548 | 52,560 | 71,975 | 48% |
| Tablets and PCs | 1,021 | 1,311 | 1,675 | 2,140 | 2,744 | 3,525 | 28% |
| M2M | 211 | 346 | 549 | 840 | 1,234 | 1,725 | 52% |
| Non-smartphones | 147 | 180 | 214 | 242 | 260 | 268 | 13% |
| Other portable devices | 0.43 | 0.36 | 0.31 | 0.29 | 0.33 | 0.38 | -3% |

Nowadays, most people consider mobile voice service as necessity, an data and video stream is becoming a daily use in consumers.

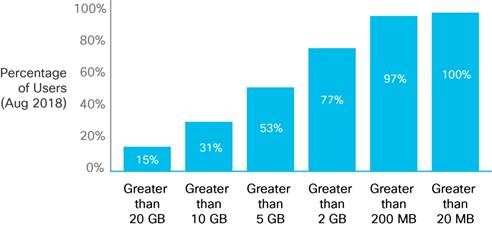
Blackhaul capacity must increase at the same speed that the demand of the consumers, and also the efficiency must improve if mobile broadband and other mobile services can effectively support the consumer usage trend.

The mobile network is evolving fast, 5G will be the leading technology for the next years. With the proliferation of mobile devices is expected and important need for networks to meet some requirements as transparent connectivity, high performance computing, enhanced real time video and multimedia, enhanced bandwidth, security, latency, uninterrupted services..

Source: Cisco VNI Mobile, 2019

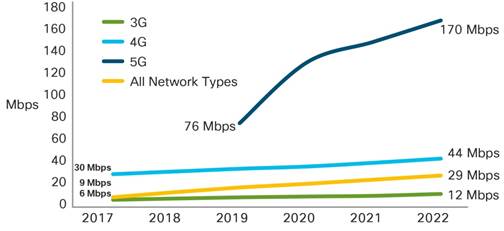
The proportion of mobile users who generated more than 2 gigabytes per month was 65 percent of users at the by September 2016, and 10 percent of the users consumed more than 10 gigabytes per month of mobile data (Figure 29) in the study. The top 10% users consume 45 GB/month as of August 2018.

**Figure**65 Percent of Mobile Users Consume More Than 2 GB per Month

[](https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.docx/_jcr_content/renditions/white-paper-c11-738429_28.jpg)

By 2022, 4G speeds will be nearly double than that of an average mobile connection. In comparison, an average mobile connection will surpass by over 2-fold over 3G speeds by 2022. Average 5G speeds will increase from 76 Mbps in 2019 to 170 Mbps by 2022. 5G is expected to be in its infancy by 2022, globally, 5G connections will be 3.4% of total mobile connections. (Figure 25).

**Figure 25.**Mobile Speeds by Technology: 2G Versus 3G Versus 4G

[](https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.docx/_jcr_content/renditions/white-paper-c11-738429_24.jpg)

However, there is a big problem that applications face which is the current centralized cloud computing architecture. If the mobile device is connected to a very distant centralized computing server, it will introduce a high latency because it tries to obtain high computing applications which will produce a high load in the RAN (Radio Access Network) and in the Blackhaul. Besides pen Mobile Network Operators (MNOs) are trying to keep up the demand of the users by increasing the cellular network capacity through the densification of the RAN.

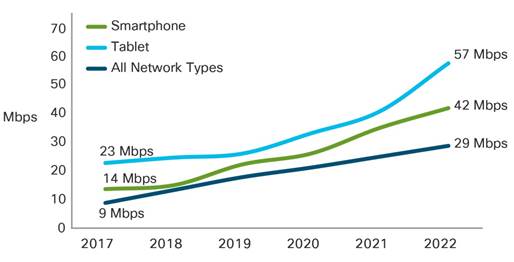
Globally, the average mobile network connection speed in 2017 was 8.7 Mbps. The average speed will grow at a CAGR of 26.7 percent, and will reach 28.5 Mbps by 2022. Smartphone speeds, generally 3G and higher, will be on par with the overall average mobile connection by 2022. Smartphone speeds will more than triple by 2022, reaching 41.6 Mbps.

The speed at which data can travel to and from a mobile device can be affected in two places: the infrastructure speed capability outside the device and the connectivity speed from the network capability inside the device (Figure 24).

These speeds are actual and modeled end-user speeds and not theoretical speeds that the devices, connection, or technology is capable of providing. Several variables affect the performance of a mobile connection: rollout of 2G, 3G, and 4G, and now 5G in various countries and regions, technology used by the cell towers, spectrum availability, terrain, signal strength, standard ratifications and number of devices sharing a cell tower. The type of application the end user uses is also an important factor. Increase in speeds by 2022 is due to the expected rollout and commercial deployment of 5G.

Download speed, upload speed, and latency characteristics vary widely depending on the type of application, be it video, radio, or instant messaging.

**Figure 24.**Mobile Speeds by Device

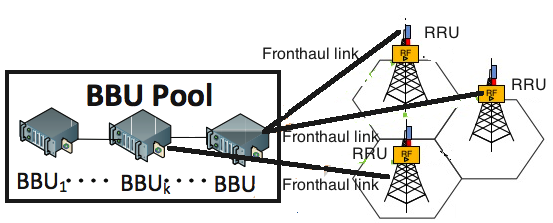
[](https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.docx/_jcr_content/renditions/white-paper-c11-738429_23.jpg)

CRAN paradigms

<https://www.cse.wustl.edu/~jain/cse574-16/ftp/cloudran/index.html>

In order to overcome this challenge, there are 2 emerging paradigms. CRAN (Cloud-based Radio Access network which the purpose is to virtualize the base station (BS) functions. This technology has a centralized architecture composed of BBUs (base-band unit) and their centralized pool, RRUs (remote-radio unit) and the transport network (fronthaul). CRAN brings the cloud computing technology by centralizing BBUs to ensure efficiently the network operation and a flexible service delivery. The virtualization of the BS can be done by NFV or SDN. The main problem is the resource allocation which includes cost, complexity, power, and energy consumption.

The other emerging solution is Mobile Edge Computing (MEC). Which provides a distributed cloud computing capabilities within the radio access network (RAN) allowing application and services to be executed in close proximity to the end users. It offer a new paradigm to liberate mobile devices from heavy computation workloads. In conventional cloud computing systems, AWS, Azure, Google Cloud Platform, this workloads are leveraged, thus incurring extra latency due to the data exchange in the Wide Area Network (WAN). However, MEC handles distributed task offloading to shift the computation to the Mobile Edge. This approximation significantly reduce the latency in the network. It also avoid congestion a prolong the battery lifetime of mobile devices. In order to accomplish this it is needed to give computing and storage capabilities to the Network Edge devices ( BSs, APs, routers…) which willl be placed close to the end users, thus reducing both time to response and end-to-end delay



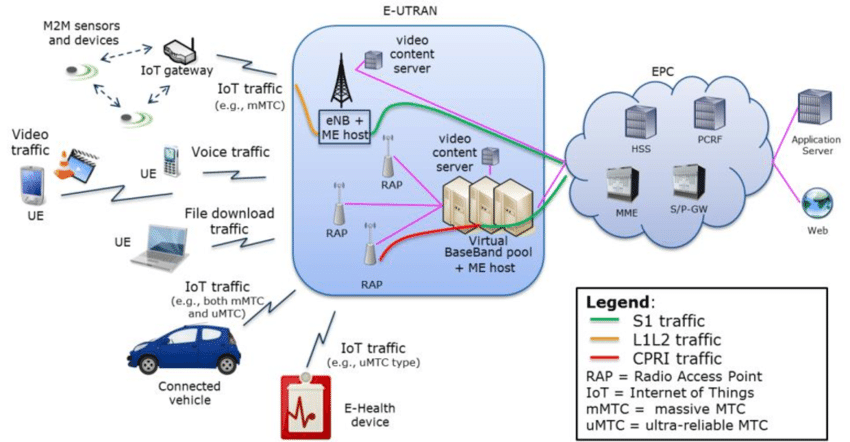
**Mobile Edge Computing (Overview and Framework)**

Mobile Edge Computing was standardized by ETSI (European Telecommunications Standards Institute) and Industry Specification Group (ISG) and acknowledged by European 5G PPP as a prime emerging technology for 5G networks. MEC can be implemented in the current 3GPP infrastructure, and it will have an important role in the 5G architecture.

MEC enables a variety of capabilities such as cloud computing, task offloading and processing tasks like content caching to run on the edge of the cellular network. This has many subsequent advantages. In the user part, it significantly improve the user Quality of Experience (QoE) providing high availability and fast response. Also, the control of the service is closer to the Mobile Network Operator (MNO) allowing the MNO to manage the user experience which bring new possibilities e.g the deploy of new applications.

The main characteristics of the MEC scenario are:

* Local computing: As MEC networks has access to local resources the can work isolated from the rest of the network which helps in security issues.
* Proximity: The proximity where is deployed give the MNO more control over the traffic and helps in the analysis of the data
* Low latency: Avoiding the need of sending the data to the core network produce higher efficiency anda ultra low delay in the response time.
* Location-awareness: Utilization of low-level signaling for discovering the location of services by the edge devices.



**Architecture**:

First we need to describe how a cellular network is formed. We have 3 main parts shown in the figure below. The Core Network, the RAN (Radio Access Network), and the end user devices. The core network is wired connected (e.g IP/ Ethernet) with the RAN. The RAN connects the the backhaul network with the BSs through a high data transfer interface. Finally, the WDs (Wirelesss devices are connected to the MON (Mobile Operator network trough the RAN, so the main task of the RAN is to provide and facilitate the connection between the red core and the user devices. It is divided by cells, each one with a BS. So the RAN is divided into several cells and it covers a wide geographical area.

Besides, BSs are connected to the Base Station Controller (BSC) or Radio Network Controller (RNC) using WLL or microwaves. These devices control the BS and carry out several functions as mobility management. In order to improve and overcome the challenge involving bandwidth and latency problems, MEC layer is proposed. This layer is situated between the end devices and the cloud. It complies with cloud computing in order to enhance the performance of mobile subscribers.

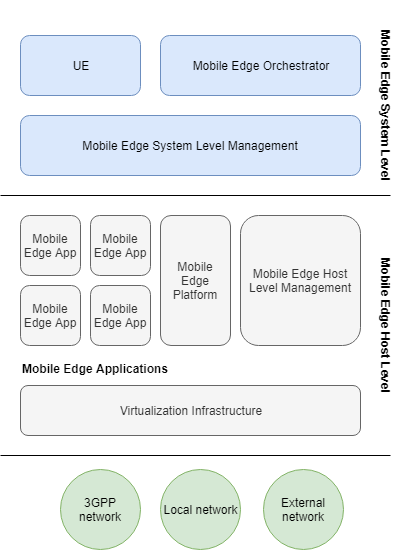
The figure below shows a complete architecture of a MEC scenario. Now we see 2 new parts, the MEC Servers, which are installed at the BSs or close to them. A MEC Server is equipped with computing capabilities. MEC architecture is locally managed by the Network Operator, not like P2P it centralized cloud servers. The generic resources of the MEC Servers are virtualized. APIs are used to expose these resources so they are accessible by the applications both the WD and Network Operator. The computation needs are served via Virtual Machines (VMs). This will significantly reduce the latency as the task is served much closer than in the cloud server.

Besides task offloading, these servers are able to provide many other tasks as content caching, traffic monitoring, location services.

**Framework architecture for MEC:**

It represents high-level functional entities grouped in MEC system level, host level, and network.

In the *mobile edge system level* we have the Mobile Edge Orchestrator which is in charge of serving and coordinating among UEs, (que es?), the mobile Edge hots, and the network operator. It also provides accounting services and maintain information on the whole mobile edge system about the topology of the deployed mobile edge hosts, the available resources in the whole edge system and in the each host, the available mobile edge services, and the applications that are instantiated. It is connected through an interface to the virtualization infrastructure providing security managing the applications by checking the authenticity and integrity, validation the policies and maintaining a catalog of the applications available.



At the mobile edge host level, the mobile edge applications runs VMs (supported by virtualization infrastructure) within the mobile edge host. At this level is done the computational execution job and the UE location. In the data plane of the virtualized infrastructure, it is executed the forwarding rules received by the the mobile edge platform and routes the traffic among the applications, services and the networks. It is also in charge of managing the virtualized resources for the mobile edge applications, this is, allocating and releasing virtualized computation, network resources and storage. Apart from managing, it provides fault tolerance and performance monitoring.

It collects and reports information of virtualized resources proving this way server and system level management information.

The mobile edge platform is the one which hosts the mobile edge services. The mobile edge applications, interacting with the platform, can advertise, discover, offer and consume mobile edge services. There is a manager which provides element management functions such as life cycle, service requirements, DNS, configuration and security. The mobile edge platform can connect with other platform via and interface for control plane procedures. Through this interface, platforms can connect an group tighter for in a grid.

**Edge deep learning framework**

We have introduce the principles of a MEC system, the framework and the architecture, however we haven’t yet discussed how deep learning can be applied to this kind of systems. We have seen that wireless devices are now big consumers of the internet, but rather consuming data only, distributed end users and terminal devices are actively contributing data to the Internet ecosystem. These data comes from different sources as text, voice, video, etc. this data is enormously valuable in terms of information worth to discover. Recent research in deep learning have shown that the analysis of this data can highly contribute in many fields, and exploring the data has great potential for video classification, speech recognition, internet advertisement… As we know, deep learning focuses on a strong computation in order to process all the incoming data. This massive computation can be provided by cluster or data centers. Besides, a cloud which relies on virtualized auto-scaled computation and scaled storage resources seems the ideal platform to develop deep learning.

In the cloud provided by Amazon (AWS), the Deep Learning AMIs (Amazon Machine Images), provide the infrastructure and the tools to the machine learning researchers. EC2 instances has pre-installed deep learning libraries as Tensorflow, to facilitate the further training of AI models.

However, still there are some issues with the cloud-based deep learning. This cloud learning performs well when the data is available at the datacenters, but, sending all the information to the datacenter will incur high traffic transmission over the Internet, which will lead into latency problems as we said before. This will challenge real time applications as face recognition or tracking cameras where the raw data can be huge and it will be highly time-consuming in both ways, transferring and training. If we go deeper, these applications meet latency requirements of ms, which seem impossible to sustain in a cloud paradigm. Even with ultra-high speed links and distributed data centers it is really difficult to meet latency requirements less than 100 ms. So, if we take into account that in the problem we are focusing, the data source will come from wireless distributions channels (WIFI, 4G, 5G), it will be even more complicated.

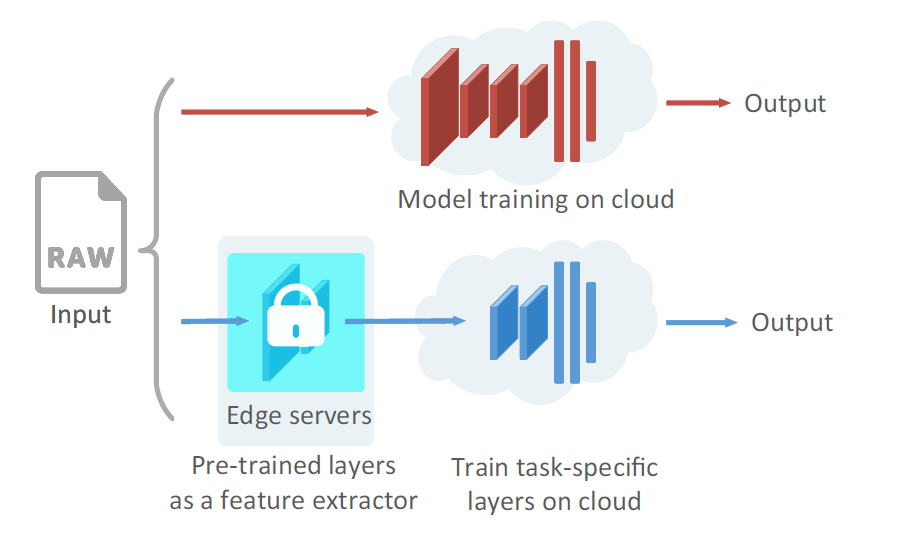
Edge computing, which has arise in the last few years, has been carried out also with the cloud computing concept. As we said before, the main task is to decentralized data center and push data, computing content, applications, away from the centralized data center and close the network edge, placing this way a significant amount of control, communication, storage, computing, management in the edge of the end users. It will derive in an important reduce of traffic to the cloud which will also accelerate the training process, giving a solution to the problem presented before. With edge computing, the latency requirements will be sustained for a wide range of deep learning applications.



**Deep learning over the network**

In the last decades, the research in machine learning has attracted many efforts from the research community. Approaches like neural networks (feed forward neural networks, Huang et al. (2004)), or recurrent neural networks. Also, multi-layer neural networks (Schmidhuber (2015)), named deep learning has also become highly used to solve a huge range of problems, e.g., image recognition, natural language processing, speech recognition, computer vision, etc. Deep learning has also gained huge success in the industry, e.g., Google, Microsoft, and NVIDIA. The 2 main things have boosted the widespread use of deep learning. Firstly, the huge amount of available data which has not been analyzed. Secondly, the efficient and powerful computing provided by parallel GPUs, which enable to use larger datasets and which really enhance the time consumed during the training.

As mentioned, deep learning relies on the availability of huge amount of data. Then, cloud-based deep learning seems a promising technique to leverage dynamic data and to provide online training. Besides, with the advance in cloud-based computing, it seems the perfect option in this era of big data computation which goes beyond terabyte scale. This not only require GPUs working, it also need really high bandwidth. This choice might be a viable way for all the data processing in deep learning techniques. It is clear that the computing power in a cloud datacenter overwhelm end user devices. However, even thought putting all the computing task in the cloud seems the perfect scenario for a high performance, it has a counterpart, as the bandwidth doesn’t enhance as fast as the data processing speed. As mentioned, users are generating a growing volume of data which is causing a bottleneck in the distribution network. Some applications as real time video analysis require tremendous network resources for the transportation and are really time consuming, even in the training part, so it Is not worthy to uploaded to the cloud. Then, the state-of-the-art of cloud-based computing is no longer suitable for applications with large data volume a transmission latency.

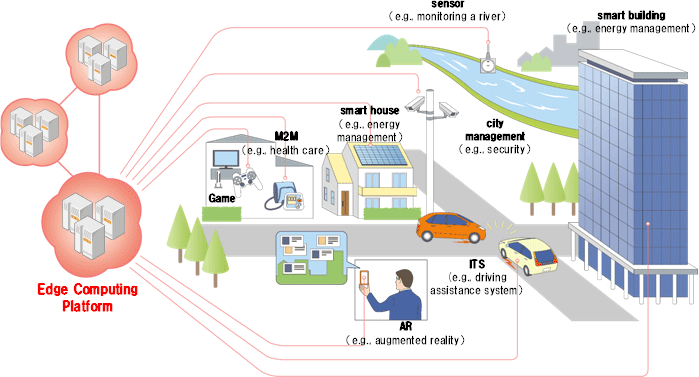


**From cloud to edge learning**

The edge network is the bridge between the end-user wireless devices and the cloud datacenters. These networks, which are directly connected to the users, have a critical role as they aggregate all the traffic from the devices and forward it to the cloud centers. The concept of edge computing refers to the technology which make possible that the computation can be done in the edge networks instead that on the cloud datacenters. In other words, bringing computation and storing resources and services from a cloud infrastructure to the proximity of the end users (edge network), thus, reducing latency. Edge learning brings possible that the computation offloading can be done on edge servers (mec servers), this will minimize the inference latency and also the volume of traffic sent to the cloud center will be lower. (bottleneck problem). It opens a new world of possibilities. Since we have now one more option. Before we could do some computation in the device itself or we could offload to the cloud. Now we can offload everything to the edge server, or also it could be possible to offload part of the data to cloud and part to the edge server. For instance, in deep learning, instead of training the whole model in the cloud, part of the data could be trained in edge servers; and the extracted features in the edge server could be delivered to the one in the cloud for further and deeper model training, and the results will be returned to end users.

**Application of MEC**

As we said earlier in the introduction, the amount of traffic each user is using is increasing significantly. If we add to the equation that the IoT devices are expected to generate a very significant amount of data in the next years as their use is becoming something ubiquitous. Actually, the amount of data traffic is not the only problem, e.g, using cloud computing with the IoT will arise some actual problems about privacy concerns and most IoT devices have power constrains (processing and memory capabilities). MEC could conform the solution by offloading part of the traffic to the edge network instead of offloading all the traffic to the cloud network. MEC could work on both channels, downstream (on behalf of cloud services) and upstream (end users)



MEC provides low latency and high bandwidth because of the proximity to the end users. It is provided to both information and computation resources for mobile access. This will provoke a decrease in the demands to the core network. The applications most suitable to be offloading in the edge are sensitive real-time ones. The objective is that the relevant data traffic will be generated, delivered to the edge network, stored, processed, analyzed and acted in the less time possible. There are many applications where this technology can be implemented.

* Offloading: This is the main one, and we will discuss a solution for this one. MEC plays a fundamental role in the desire need of the increasing of aspects such as unlimited computing, storage, high bandwidth, allowing more and more complex applications to run in the wireless devices of the end users. Besides, reducing the power consumed by the batteries is another goal to achieve. First of all, this tasks are offloaded either because it is wanted to reduce the power consumption or because the tasks are too complex to be executed in the WD, limited hardware capabilities. In the case that there are no more MEC servers available, it will be possible to offload to the cloud servers, to a farther MEC server or perform the task in the device itself. MEC is a really convenient approach to offload this heavy computation, accelerating the processing time. Both the energy consumption and and the device computational capability will no longer be a bottleneck in the execution of jobs like high speed browser, sensor data processing, video analysis, 3D rendering, language translation.
* Surveillance and Safety: Pattern recognition and video surveillance are applications that require and intensive computational load. MEC can play an important role as can provide low communication delays to fulfill real-time automated surveillance requirements. An example of an scenario could be a local network of of IoT devices as video-cameras connected to a broadband mobile network through an IoT gateway. The video stream will be send to the mobile edge host, in which the surveillance app is running. In the host it will be also carried out the real-time processing. The advantages of this architecture are e.g if there is an error during the execution, an alarm can be trigger and send straight to a central office which will be located accessible in the Internet. Another advantage is that this configuration is cheaper compared to dedicated equipment and infrastructure by leveraging the RAN and MEC system of the network operator.
* Augmented reality, virtual reality and gaming: All these 3 applications have very strict requirements of low latency and high computational jobs, which can overwhelm the computation capabilities of WDs. Again, the offloading of these jobs in to a MEC server can bring many benefits creating a balance proximity, thus reducing delay and extending the power life of the device.
* E-care: MEC can also be part of this more a more developed field. The exponential growth of human-computer devices (smart bands, smart phones..) in bringing a new whole case-scenario where it is fundamental the reliability of the system. It enables devices to collect all kind of data like temperature, pulse rate… and send it to the server for storage, analysis, sharing… In a medical case it will allow the doctors to access patients information in real-time and accordingly diagnose them.
* Moving devices and smart vehicles: As MEC role is offering local processing and low delay, it can be very useful to mobility use cases, which is expected to be important to 5G services. The ore network is in charge of mobility ad session management. In the current 3G and 4G networks is performed by the serving gateway (SGW) and serving GPRS support node (SGSN). However, IoT applications will not require complex traffic filtering and manipulation, so mobile edge hosts would be able to carry out consistent control plane operations, e.g., mobility, QoS enforcement, charging, etc. Thus, it will create a distributed environment synced with the local sensors which will be scalable reliable.
* Sensors: Today, a tremendous number of smart devices and objects are embedded with sensors, enabling them to sense real-time information from the environment. This phenomenon has culminated in the intriguing concept of the Internet of Things (IoT) in which all smart things, such as smart cars, wearable devices, laptops, sensors, and industrial and utility components, are connected via a network of networks and empowered with data analytics. In the past few years, many startups have embraced and actualized the concept of IoT in areas including smart homes/buildings, smart cities, smart environments, and so on



Now, we will describe the threats that the MEC technology needs to face for a correct implementation.

* Infrastructure: The deployment of real-time data processing brings several opportunities of malicious attacks as the concept of edge means that it involves the last mile of the network. Attacks like DoS and jamming which will consume the bandwidth and the computing resources will lead to a poor performance. Besides, one malicious user could compromise the correct functionality and take control over the network.
* Virtualization: The concept of virtualization is based in share resources in the edge environment. There are many security issues that need to be ensure as the proper protection of the APIs, they should be protected against malicious activities as is the gate to deliver the information. Network Function Virtualization (NFV) is used as the virtualization technology which consist in a dynamic and flexible provisioning of services and network functions using generic hardware. So the hardware installed in the base stations must require some requirements of computation in order to enhance the performance compare to dedicated hardware.
* Mobility: The handoff between base stations is one of the main concerns in cellular network. In MEC environment it is needed that the active applications running on both the server and the mobile hosts should keep working as before the handoff. This process needs to meet requirements as the communication between BSs and the handoff itself. In order to achieve this, it will be needed a high number of BSs with MEC servers running so the computation for the handoff will be more easily carried out.
* Discovering and computing resources: In this scenario, MEC provides the mechanisms need for the search and discovery of the nodes involved. This task can be complicated due to the high number of devices that are part of the edge network. It must be achieved a fast interaction of all the resources in a transparent way. Also, dealing with the availability and reliability of the node involved can be very challenging. Ensuring a proper functionality will lead us to te second step which is adding in a correct manner the computing resources to be able to analyze and process the data tasks. Resources which should be flexible and portable through different types of environments.

In conclusion, with the rapid proliferation of wireless devices, including mobile phones, wearables, cars, sensors, the wireless traffic has grown dramatically causing potential congestion in the core network and causing significant strains in the blackhaul bandwidth. If we sum the emergence of brand-new services and application as augmented and virtual reality, gaming, surveillance, intellect transportation which requires highly efficient real-time data collection and processing. To solve these problems and challenges, MEC is born to provision storage and computation resources to end users in proximity. It will address the fact that demands can now be served quickly and locally rather than in a remote cloud server. Consequently, it proposes a decentralization of the cloud service moving the computation burden from cloud severs in the core network to the base stations in proximity to the users. Thus, enhancing both the request of contents the computation offloading. It will dramatically mitigate the pressure in the core network and reduce the service latency

## STATE OF THE ART

A brief introduction of the State of the Art is given in this part, it will be divided in two parts. First it will be discussed LSTM–state of the art and the second part will be focused in how deep deep learning is being applied to Mobile Edge Computing instead of giving the actual state of MEC implementation in the current situation.

**LSTM**

During the last decades, time series prediction has been a field of study, the Autoregressive Integrated Moving Average (ARIMA) [31] model has been used to study time-varying processes in statistical signal processing. However, it has a limitation, its performance decrease when it comes to rapidly chasing data as it has a tendency in concentration in the mean values of the data series. Another technique used for time series prediction is Support Vector Regression (SVR) [32]. Even though it has been successfully applied, it also has a limitation which is a lack of structured means to determine key parameters of the model.

Today, research and applications of LSTM for time series prediction are proliferating. For example, Wang et al. [33] used LSTM-based model to predict the next moment traffic load in a specific geometric area and Alahi et al. [34] predicted the motion dynamics in crowded scenes based on LSTM. In [38], they study the mobile traffic of an LTE base station and predict the traffic using RNNs.

Geant Network [35]

**Deep Learning**

Data analytics capabilities can be done by cloud-centric processing techniques, however, when it is required ultra-low latency for the processing, a distributed mobile edge computing architecture can enhance the performance as the capability of smart mobile devices is increasing. It can be applied different deep learning techniques for different aspects which will be discussed next such as *automate detection process, detection of security threats, task offloading decision, content caching, reliable mobile sensing, etc.*

Many research efforts have been carried out in the MEC context. Different low-complexity algorithms are proposed to solve the binary computation offloading problem in the literature [4–12]. A distributed algorithm based on game theory is proposed for MEC system in [1], which requires multiple iterations of communications between the edge server and WDs. Another iterative algorithm to solve joint task offloading and resource allocation in MEC networks is to iteratively update the binary offloading decision [2,3], where the traditional resource allocation problem is solved when the binary offloading decision is present. By relaxing the binary constraints to real variables, [4] proposes an eDors algorithm for MEC systems. The algorithm is further extended in [5], which iteratively improves the binary offloading decisions after relaxation. Separable semidefinite relaxation is applied to jointly optimize the binary offloading decisions of all users and the communication bandwidth allocation in [6], whose complexity is still too high for real-time offloading. However, all those algorithms are limited by the trade-off between optimality and computational complexity, which is not applicable for real-time computing offloading under the MEC networks with time-varying environments. Deep learning that uses a deep neural network (DNN) with multiple processing layers to learn representations of data has achieved many breakthroughs in different areas, e.g., robot control [7,14], natural language process [8], and gaming [9]. For systems with large-scale state-action space, using a DNN to approximate the state-action relationship can obtain near-optimal performance [9,10]. Deep learning has also been applied to solve computationally expensive problems in wireless communications [1 1,18], e.g., resource allocation [12, 13], signal detections [14 15], interference alignment [16], and *caching* [17]. There exist few recent works on deep reinforcement learning-based offloading for MEC networks [18–21]. However, most of existing works for MEC networks are based on the deep Q-network, whose tabular-search nature is not suitable for handling problems with high dimensional space. There is also some work in joint offloading decision and resource bandwidth allocation in the last years, e.g., using a prediction model and cross entropy (CE) method for the offloading strategy [29]. Another similar work in [30], but in cloud edge rather in mobile edge. However the edge server considered in MEC network is performed by a Computing Access Point (CAP)

Some other work done in edge computing are, e.g., edge computation offloading Tan et al. (2017) [22] Li et al. (2017) [23] Chen et al. (2016) [24] Habak et al. (2015) [25], edge caching Drolia et al. (2017) [26], and edge resource allocation Wang et al. (2017) [27]. The edge computing devices can be any computing or networking resource residing between data sources and cloud-based data centers, e.g., a 5G cellular tower between smartphones and the cloud infrastructures. Deep learning has achieved great success in numerous applications. A distributed deep neural network (DDNN) Teerapittayanon et al. (2017) [28] is proposed to partition neural networks across mobile devices, edges, and cloud, so as to reduce the latency

**TOOLS USED**

**GOOGLE COLAB**

Colaboratory is a free Jupyter Notebook environment that requires no configuration and runs completely in the cloud, it Is accessible form a web broswer. It allows to develop deep learning applications on the GPU for free. Besides it has some important features as:

* Collaboration (sharing code easily)
* No need to install anything
* Free GPU (Tesla K80)
* Supports Python 2/3,
* Preinstall libraries used in AI as tensorflow, keras, etc.
* Google Drive support.
* It can be connected to Github.



**PYTORCH**

It is a Python package designed to perform numerical calculations using the programming of tensors. It also allows its execution in GPU to speed up the calculations. Normally PyTorch is used both to replace numpy and process calculations in GPU and for research and development in the field of machine learning, mainly focused on the development of neural networks.

Alternatives to PyTorch:

There are several alternatives to PyTorch in machine learning, some of the best known are:

* Tensorflow: was developed by Google Brain Team. It is free software designed for numerical computation by graphs.
* Caffe: is a machine learning framework designed to be used in computer vision or image classification. Caffe is popular for its library of models already trained (Model Zoo) that do not require any extra implementation.
* Microsoft CNTK: is the free software framework developed by Microsoft. It is very popular in the area of ​​speech recognition although it can also be used for other fields such as text and images.
* Theano: is a python library that allows you to define, optimize and evaluate mathematical expressions that involve calculations with multidimensional arrays efficiently.
* Keras: is a high level API for the development of neural networks written in Python. It uses other libraries internally such as Tensorflow, CNTK and Theano. It was developed with the purpose of facilitating and speeding up the development and experimentation with neural networks.

PyTorch is really recent library and despite this it has a large number of manuals and tutorials. In addition to a community that grows by leaps and bounds. PyTorch’s interface is simple and allows easily to create neural networks despite working directly with tensors without the need for a library at a higher level such as Keras for Tensorflow.

Unlike other packages such as Tensorflow, PyTorch works with dynamic graphs instead of static graphs. This means that at the time of execution the functions can be modified and the calculation of the gradient will vary with them. However, in Tensorflow we must first define the computation graph and then use the session to calculate the results of the tensors, this makes it difficult to debug the code and makes its implementation more tedious.

Besides, PyTorch has support for its execution in graphics cards (GPU), uses CUDA internally, an API that connects the CPU with the GPU that has been developed by NVIDIA.



**TENSORFLOW**

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015

TensorFlow offers multiple levels of abstraction. Keras is the high-level Keras API, which makes getting started with TensorFlow and machine learning easier.



## LSTM

Traffic prediction is becoming a fundamental task is new the cellular network generation. It is needed to provide an optimal resource allocation to predict the user demands on the network. Besides, the dynamics of the network represent a significant issue for this task. There is a huge massification of the wireless network which make it more challenge. Also, the different number or wireless devices demanding traffic and the different kind of services they require bring the need to an enhance in terms of capacity, latency and performance. These requirements require the network to be aware in advance of the traffic demands. Traffic forecasting and traffic analysis can help in this task making the network more intelligent, allowing a enhance resource management. This will derive to a smart optimization of physical resources which will improve both, the QoE of the end users and energy consumption.

In the last decade, the development of powerful hardware has unleashed a variety of new technologies as machine learning. This technique will allow to execute traffic prediction algorithms with minimal efforts and with excellent results. In order to train these neural networks, it is need datasets with real cellular data, however, from the research perspective, there is a lack of datasets related to cellular networks. Network Operator don’t usually make available the user data traffic. It can be found but very limited, just aggregated traffic from CDR (Call Detail Records), where the is no information about the base station users are connected or separate traffic between data, text or voice.

The first part of the scenario has been done using Time Series Prediction, this concept can be described as the extraction of useful information from past records determining this way future values. However there exists many obstacles as learning long-range dependencies from the data. Most deep learning algorithms don’t success in this task, although Long-Short Term Memory (LSTM) surge as a promising solution to effectively overcome the problem.

We are going to use LSTM to create a deep learning model to predict traffic in a cellular network and use this prediction to feed a neural network to perform offloading decision.

**Time Series Prediction**

In the telecommunication field, time series prediction has been used to analyze several numbers of time-dependent events, e.g data traffic, user location, requests from users, etc. Besides, guaranteeing an acceptable QoS and QoE regardless the dynamics of the network is a priority issue that need to solved. The prediction of the data traffic the user will request and the topology location of the user are key values to efficiently serve the demands of the users. It will be helpful since network operator can reserve resources accordingly. This reservation can lead to several problems if the predictions are not correct, not only it will reduce significantly the QoS, but it will decease completely the overall performance. Therefore, it is quite important to ensure that the prediction results are accurate.

Time prediction consists in the estimation of a value at a time t, based on its previous data . Let denote with , the main goal resides in estimate values of the function so that will as close as the real values (ground truth).

Nowadays, deep learning model are becoming more useful in time series prediction. Recurrent Neural Networks are playing a fundamental role, specifically a kind of RNN called Lon Short Term Memory. LSTM are used because the overcome the problem of the vanishing and exploding gradient that normal RNN face in time series. This is due to the problem that the influence of the input in the hidden layers of the networks either will vanished or will increase exponentially causing the network not to achieve the expected behavior.

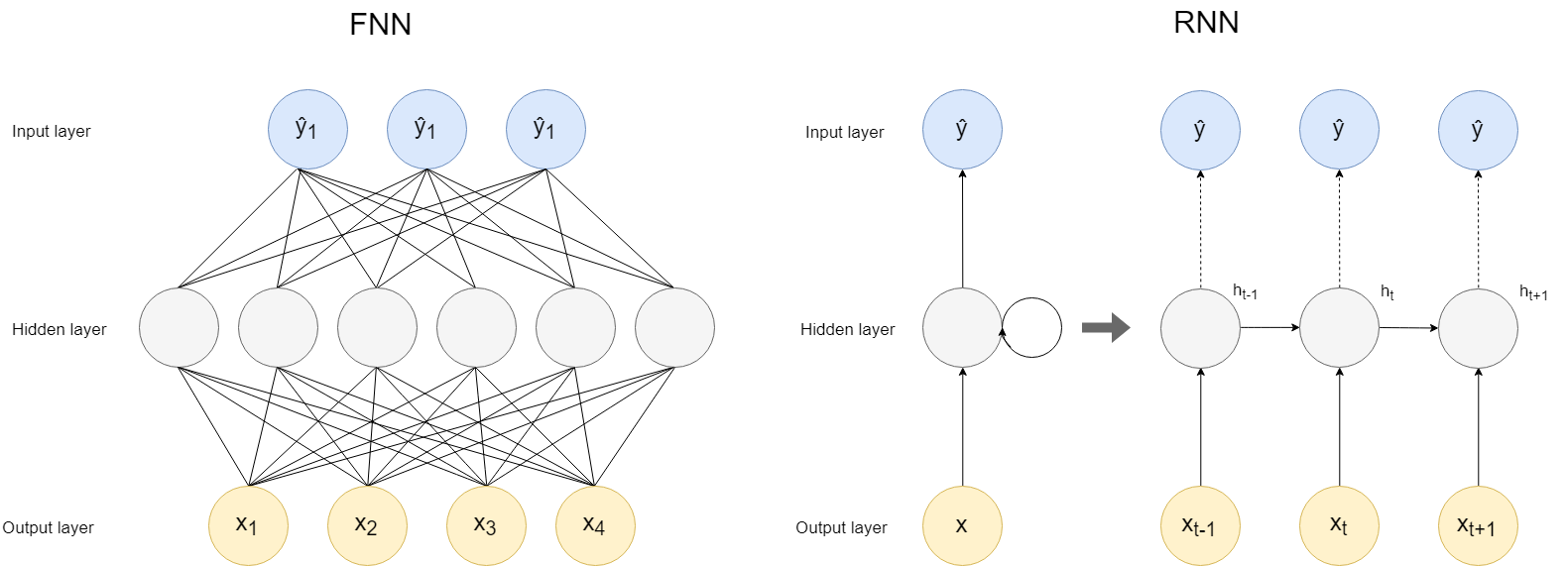
LSTM can achieve this task modifying the structure of the hidden layers, controlling this way how are influenced during the cycling connections. The data used for this application is traffic data from GEANT networks, a European research network. The prediction of traffic data in the network is of great importance in the optimization deign of the telecommunication network and it will help us in our job of producing a task offloading algorithm in the MEC scenario.

OVERVIEW

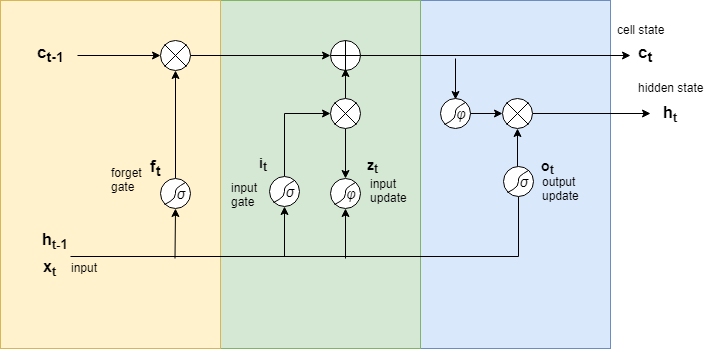
Artificial Neural Network (ANNs) in a brief way is a machine learning model which eliminate the disadvantage of the traditional learning algorithm based in rules. There are 2 main models inside ANN.

Firstly, Feed Forward Neural Models (FFNN) which consists only in 3 layers (input, hidden and output). Each layer is composed by a number of neurons and the activation function. The main feature of this model is that there is no connection between the neuron of the same layer and that the information flows in one direction from the input to the output layer passing through the hidden layer if any. There are many fields where FFNN are used as classification, object recognition, image processing, etc. Although they cannot be used to treat data time-dependent as our problem state.

Secondly, Recurrent Neural Networks (RNN) with a very similar structure but that differs from FFNN because it allows the connection between neurons of the same layer. RRNNs calculate the output of the current moment taking into account the input and the hidden state of the previous moment . So, this kind of ANN indeed allow to use historical data from the input to the final output. But as we said earlier, even though theoretically RNNs can handle long-range dependencies, the problem of the vanishing gradient makes it unable to perform the task. This is explained in this paper *(S. Hochreiter and J. Schmidhuber, “Long Short-TermMemory,” Neural Computation, vol. 9, no. 8, 1997, pp.1735–17800).*



In order to overcome this challenge, LSTM are brought into scene. This special RNNs is suitable for managing long-dependencies. The part that make this possible is named memory-block. It is composed by the memory cell which has information about the cell state and the gates. There are 3 types of gates, input, forget and output gates. The gates are responsible of controlling how much information flows. According to this, input gates control how much information goes into the memory cell. Forget gates control how much information remains in the current memory cell. And the output gate control how much information is used to compute the output activation of the memory block, which will further flow in to the neural network.



Let’s review some concept about how LSTM network works. In first place, the network needs to decide which information should forget of the cell state. This is done by the forget gate. The activation function used in this kind of gate is called the sigmoid function

This activation function has a range of returned values between 0 to 1. So, it is used this function because 1 will represent ‘completely save the information’ whereas 0 means ‘completely forget it’.

The second step is to decide what new information the network is going to store in the cell state. This step can be divided in 2 parts. First, a sigmoid layer (input layer) will decide which values we will update. After, a tanh layer, which is uses a tanh activation function

,

which is another kind of activation function that outputs in a range between -1 and 1, it will create a vector of candidates values , that could be added to the state. Thus, we will use these two to create the update to the cell state.

So, we have the old cell state represented by Ct-1, we will multiply Ct-1 (old state) by ft which is the output of the forget layer with the decision about forgetting. After, we calculate the convolution between it and Ct, as to say, we are computing the vector of candidates scaled by how much we want to update.

Finally, it is needed to decide how is going to be the output. It will be the information stored in the cell state but after applying a filter. First, a sigmoid layer which will decide which parts of the cell state will be output. After, we pass the cell state through a tanh which will be multiplied by the output of the sigmoid layer, this way, we will decide what we want to output .

How to ensure that the data we are using make sense:

We suppose that the traffic from the link 1 – 2 is the total traffic send by 3 users to the access point.

For instance, 20 MB which will be divide by 3 makes 7 MB per user. Some users will be using more than the other user at a time t.

We use this data calculator:

<https://www.sasktel.com/wps/wcm/connect/content/home/tools/datacalculator>

## OBJETIVES

…

# desarrollo

Esto es un ejemplo de cita a una referencia bibliográfica [1]…

## TRAFFIC PREDICTION

As we said in the introduction the first task will be the traffic prediction. In order to accomplish the task, an LSTM network will be used, the output data from this part will be used to feed a neural network to perform offloading decision.

So, the first task we realized was the pre-processing of the data. As we said earlier, we have used traffic data from a link in the GEANT network. [1]

[1] Steve Uhlig, Bruno Quoitin, Jean Lepropre, Simon Balon Providing Public Intradomain Traffic Matrices to theResearch Community

This dataset consists of traffic matrices built using full IGP routing information, sampled Netflow data and BGP routing information of the GÉANT network, one per 15 minutes interval for several months.

**Conversion from XML to CVS**

The data is downloaded in XML format, so it has been completed a conversion to .csv format to make easier treating with the data in further uses. The raw data is in kbps and is sampled during 4 month every 15 minutes. In order to convert from XML to csv matrix traffic it is used the library *xml.etree.cElementTree.*



Then, each document is parsed and using the attributes IntraTM, src and dst, the data is extracted and saved into a (23,23) matrix. One matrix per each XML file. (15 minutes pooling).

**Saving data into npy file.**

Now that we have the data saved into the CSV file, it is need to perform 2 tasks.

* Choose the data that we really want to use to train the neural network.
* Save the data into a npy file.

So first, we convert the all the csv matrices (10772 files) into one single numpy file. (traffic matrices). Besides it is also saved the datetime of each file (year, hour month, day and hour, if required)



Secondly, it is chosen the traffic between link 1-2 and save it to another file (traffic 1-2). This file contains 10772 samples of the traffic between both links.

**Preprocessing of data**

The preprocessing of the data is divided into normalization and preparing the data for feeding the network. The normalization will help the network during the training phase, e.g., to converge faster.

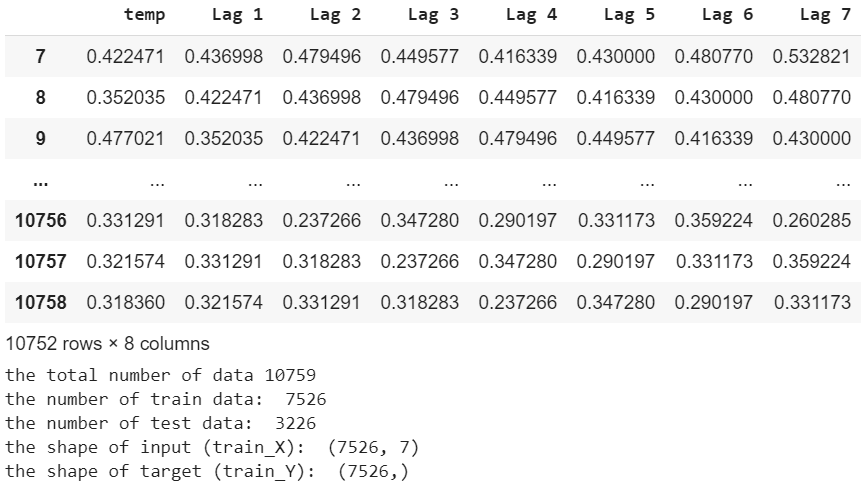
The normalization is divided in 2 steps:

1. Take the base 10 logarithm of all the values.
2. Use the max-min scaling normalization which will output our data into values between [0,1]

After normalizing the data, it is needed to prepare it. Real time prediction requires sliding window which consist in the shift of the data into a fixed number of time slots, it will feed the network with continuous data.

This is formally called as *lags* which are very useful in forecasting models because of the autoregression, described as the tendency for the values within a time series to be correlated with previous copies of itself. It will help to identify patterns in time series analysis which derives in the possibility of determining seasonality (tendency to repeat at periodic frequencies).

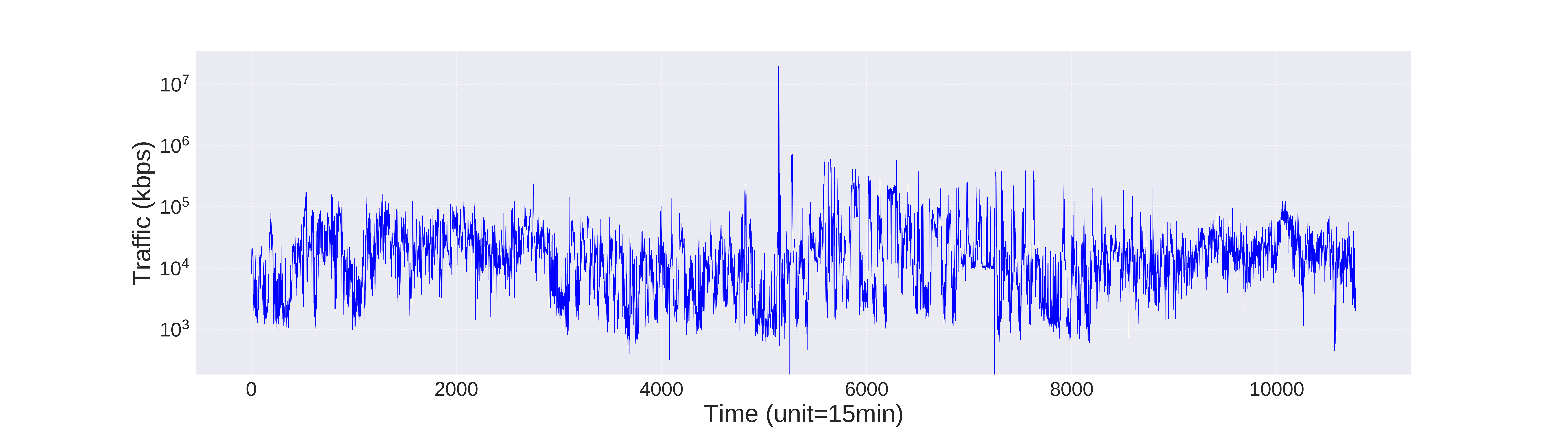
Finally, we split data into training and test set. The data is divided 70% for training and 30% for the test part.



The evaluation metric used is Root Mean Squared Error (RMSE), with this metric we will evaluate the prediction accuracy. This is used because it measures the difference between the real value and the predicted ones. The expression of RMSE is:

Theoretically, RMSE measures the square root of the mean of the deviation squares.

Here we can see the plot of the traffic before the normalization.

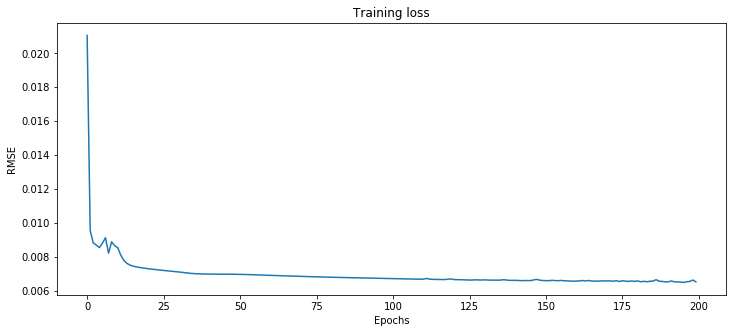


The hyperparameters of the network are described in the next table:



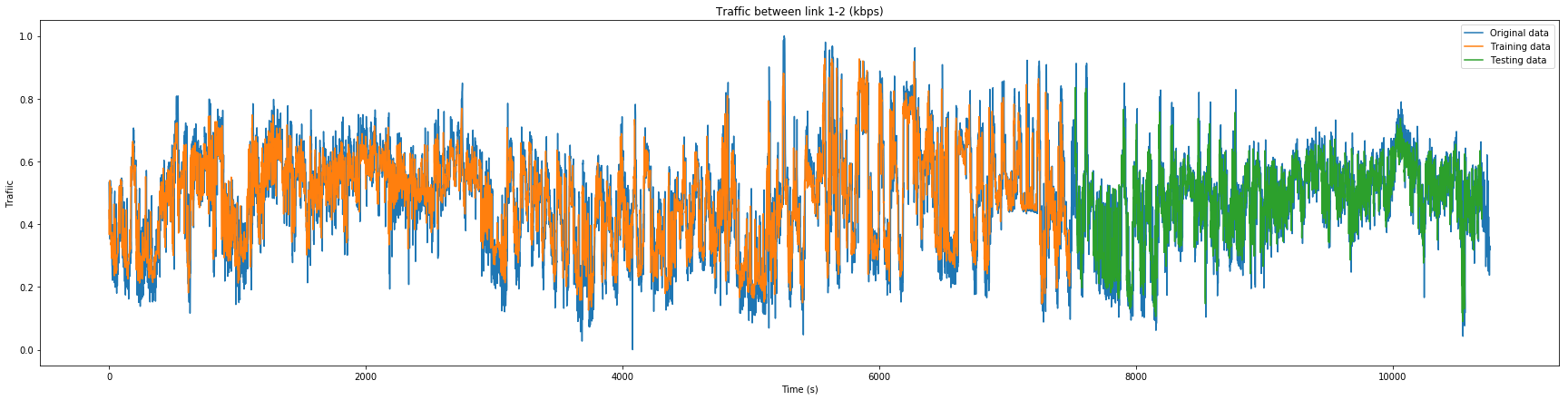
The neural network is created using the library TensorFlow. (Hablar un poco de la red y meter captura de las epochs)

We can observe how the Root Mean Square Error loss decrease with the number of epochs. It means the network is learning as the difference between the real and predicted values is decreasing.

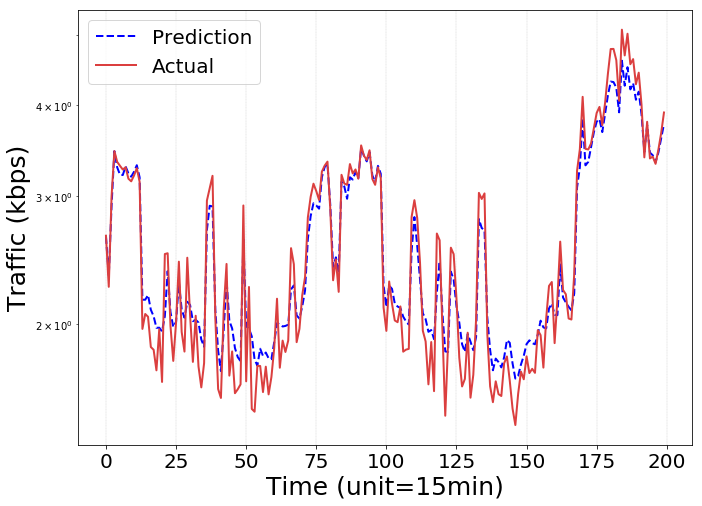


In the next figure we plot the actual traffic and the predicted one to conclude that the network is working correctly and the prediction is accurate.

For the whole dataset the predicted traffic is as follow:



In the next figure we can see a sample of traffic more detailed:

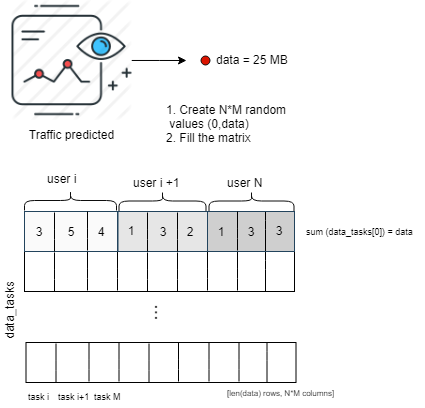


Besides, the model can be saved for future processing.

Now that we have predicted the data, we move to the second step in our job which is the task offloading in MEC. We will use the prediction to anticipate changes in the network and allow the Network Operators to reserve resources accordingly. Also, the predicted will serve as the input of offloading decision system. As it was referenced before, it is supposed that the traffic in the link 1 – 2 represents the traffic of 3 users (wireless devices) connected to an AP (access point) which is connected to the base station where the users are attached.

The feeding of the network is as follow:

1. The data (traffic) is divided into an array of size m\*n where m represents the number of tasks and n the number of users
2. It is used a random distribution to fill the values of a matrix data\_tasks. It will represent in terms of data load, the data task each user needs to offload.
3. Data\_tasks is filled by random numbers between 0 and the data traffic at time t and it has to be accomplish that the sum of all the values in the array is the traffic predicted value.



## TASK OFFLOADING

We are going to study the case in Mobile Edge Computing where multiple wireless devices WDs want to offload the task into a MEC Server. It will be carefully studied the decision of a WD to offload the computation task or not. In the case that many devices want to offload the task at the same time is going to produce a big congestion in the uplink wireless channel, thus producing a severe delay in the completion time in the tasks. This problem will cause a full degradation in the MEC performance. It will destroy the reason why MEC is useful. Therefore, in order to obtain smart task offloading decisions is highly important to take into account the subsequent problem of bandwidth allocation. In order to solve the bandwidth allocation, we will use the predictions that were computed before. This will further improve the task offloading as it will make it a more realistic scenario.

It is proposed a deep learning-based offloading algorithm to reproduce a framework which will produce offloading decisions in a MEC environment in an effective and efficient way. Deep learning has been applied to wireless communications problems as resource allocation, content caching, offloading decisions, etc.

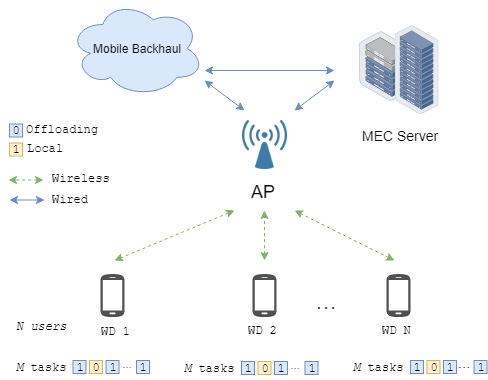
In the proposed scenario we consider a MEC Server and N wireless devices (WDs), besides, any WD will have M tasks to compute, the WD can offload the task into the MEC Server or compute locally. In the algorithm we will have 3 networks (offloading actors) producing offloading decisions. Besides, each of the network will have a memory where the decisions will be stored. The data in the memories will be used to train the networks.

ARCHITECTURE.

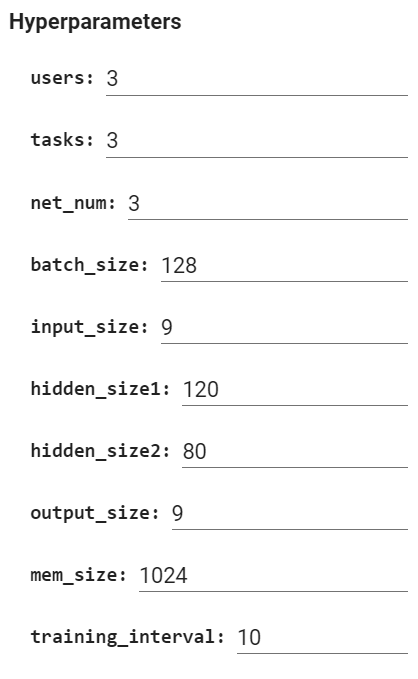
In our scenario we consider a MEC network formed by only 1 MEC server (in the edge of the network), one access point AP and multiples WDs. We are not taking into account the cloud core network in offloading terms, we only focus in the possibility of performing offloading to a MEC server or locally, so the main task is to avoid the backhaul traffic generation to remote data centers by deploying computation servers at the user side.

The WDs are connected to the MEC server by the AP and each one has M tasks which can be processed locally or can be of loaded to the MEC Server. Each task is described by a task size which will define the workload of the m-task of the n-user. It will be defined by w\_mn. Besides, another variable is defined for the offloading decision.

Denote as the binary variable, so if the user decides to offload the task m into the MEC server, or if the WD decide to compute the task locally.



The hyperparameters of the network are as follow. We use 2 hidden layers and the input and output dimension has size (users\*tasks), as it is required to obtain an offloading decision from each task of each user



It is not needed any normalizing of the data as we are using the normalized data from the traffic prediction. The network consists in 3 similar networks, each one will produce it own decisions. The last layer of the network is a sigmoid layer so the output of the network will vary from 0 to 1. In order to translate this data in to meaning data, if the output > 0.5, it will be placed a 1, otherwise it will 0 the offloading decision.

**SYSTEM MODEL**

We neglect the transmission delay from the AP to the MEC server as we suppose is connected through optic fiber so the delay compare to the wireless channel is negligible. We also neglect the delay of the downlink, the computing results from the server to the WDs because this data size is smaller than in the uplink. Besides, we assume that the server only starts processing when the task is fully received and can only send computing response when is completely computed.

We are going to focus in reducing the system utility of the system as the problem formulation which depends on two variables: energy consumption and delay. Therefore, it is needed to compute both in the case of local offloading and edge offloading.

***Edge processing***

The energy consumed by the wireless device when it is performed the uploading of the workload to the MEC server is denoted by . The energy cost is modeled following a linear dependency with the workload . Let’s denote the total cost for user n for uploading the task m to the MEC server as: This cost includes the cost of sending the data to the MEC server and computing the task.

And the total cost when the user executes MEC is given by:

* whereis the weight of the energy consumption at the MEC server, and is edge energy consumption per data bit.

***Edge delay***

The delay in computation offloading is formed by the sum of two delays. First, the transmission delay when the user n transmits the task m to the MEC server is given by:

Where the is the allocated bandwidth for the user n when he wants to transmit the task to the MEC server. Secondly, the processing delay in the MEC server which is the time the server spends computing the task offloaded by the user. It is given by:

(Here it multiplies with the encoder (cycles/s) so the output is seconds.

Where *fc* is the server processing rate. So, the total delay of the user n when he decides to offload the tasks to MEC server is given by.

Thus, if the offloading decision is 0, the delay will be zero, and if the decision is 1 (offloading to the mec server), the total delay is the sum of both, transmission and processing delays.

***Local processing:***

Next step is to declare the energy consumption in the case the wireless device decides not to offload the workload. In this case, denoting the local energy consumption per data bit as . Thus, the energy consumption of user n processing its task m locally is given by:

And the total energy consumption when the offloading decision is local is denoted by:

***Local delay:***

This is simpler as we only need to take into account the local processing rate denoted by . Therefore, the processing time when user n executes locally the task m is given by:

Then, the total local delay for a given user is given by:

**PROBLEM FORMULATION**

In order to solve this task offloading and bandwidth allocation problem, it is calculated the system utility defined as the weighted sum of energy consumption and task completion delay, as:

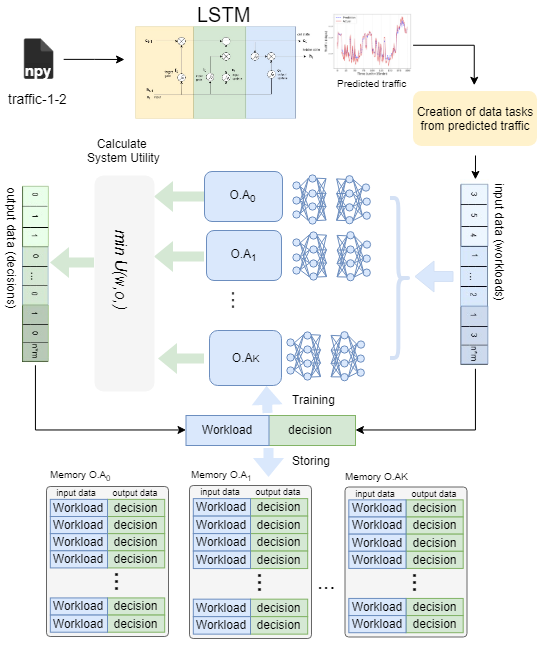
Therefore, we can formulate now the first problem which is minimizing the system utility by jointly optimizing the users offloading decisions and the bandwidth allocations. It is expressed as follows:

We observe that we have three constraints; First, the sum of all the allocate bandwidth must be less that the total bandwidth C, and that it must be zero or positive. The last restriction comes from the binary offloading decision.

A table with all the notations is presented:

|  |  |
| --- | --- |
| Notation | Definition |
|  | Data size of task *m* of user *n* |
|  | Offloading decision. if user *n* decides to offload data task *m* to MEC server. if user n decides to process locally |
|  | Transmission energy consumption of user *n* sending task m to the MEC server |
|  | Server edge energy consumption per data bit |
|  | Total cost for user *n* for uploading the task *m* to the MEC server |
|  | Transmission delay when the user *n* transmits the task *m* to the MEC server |
|  | Allocated bandwidth for the user *n* when he wants to compute edge |
|  | Total bandwidth |
|  | Edge server processing rate |
|  | Edge processing delay when user *n* sends the task *m* to the MEC server |
|  | Total delay of the user *n* when he decides to compute MEC |
|  | Local energy consumption per data bit |
|  | Energy consumption of user *n* processing its task *m* locally |
|  | Total energy consumption when the offloading decision is local |
|  | Local processing rate |
|  | Processing time when user *n* executes locally its task *m* |
|  | Total local delay of user *n* |
|  | System Utility |
|  | Weight between energy consumption and delay in the system utility |

It is proposed an algorithm to efficiently solve this problem:



It is built k parallel neural networks in the model which will produce offloading decisions. The algorithm created for MEC task offloading is composed of 2 parts, first the offloading action generator and the learning part. The system works as follows:

At the beginning, the input from the data\_tasks file created from the predicted traffic, is feed to the K offloading actors which will produce offloading decisions. For instance, if the number of networks is 3, for an input of workloads, the output will be 3 candidates of offloading decisions.

Next, it will be calculated for each offloading decision, the system utility as we studied before. However, there is a remain problem, a bandwidth problem, it is need to determine the available bandwidth for each user in the system.

Let’s denote the sum of all the tasks each user has to complete as . This information is known in advance as this is predicted traffic, so Network Operators could use this information to allocate more bandwidth to the user n in particular.

Therefore, the available bandwidth for each user n will be:

Where C is the total bandwidth.

Therefore, after calculating the system utility, the offloading decision with lower system utility is chosen among them. This will be the output of the system.

Next part is taking this output and join the input (workload) and feed this tuple of values to a memory. Each network will have its own memory. Thus, it will exist K memories in the system. This will be used to further train the offloading actors, which is actually a neural network which after being trained, it will take almost optimal offloading decisions. Each NN will have the same structure but different parameters values.

The process of producing offloading decisions and filling up the memories can be seen in the next figure:



At this point we have generated and chose the offloading decision. The data used to train the k neural networks will take a tuple of new entry labeled data (w,o) as we said before.

The memory will have a size of 1024 rows x 18 columns for the workload w and de decisions o. The NNs will extracts randomly batch of data from its own memory. It will be extracted in batches of size 128. Therefore, it is applied a gradient descent algorithm for minimizing the cross-entropy loss of each NN. The system loss will be the sum of the losses of each NN.

As we said, the memory has a finite size, the new data will replace old data. This will significantly improve the data efficiency and the computation speed. This is because newly generated data will be more preferred as it is randomly extracted.

The algorithm proposed is implemented in Pytorch. In the first stages of the training, the offloading decision will not be optimal as the memory is still filing up and the parameters of the K NNs will be randomly initialized.

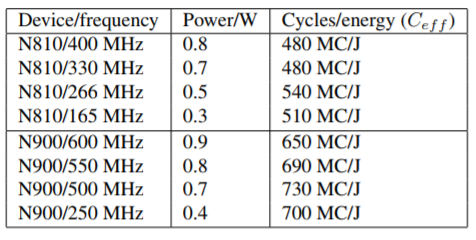
*Algorithm*

1. **Input:** Workloads from LSTM traffic prediction at time t
2. **Output**: Optimal offloading decision (lower System Utility)
3. **Initialization:**
   1. Empty memories
   2. Initialize with random parameters.
4. **for t=1,2… do**
   1. Feed the K NNs with the workloads
   2. Solve the bandwidth allocation problem.
   3. Generate the k-th offloading action candidates.
   4. Choose the best candidate as the output width lower delay.
   5. Store the new entry labeled data into the memories
   6. Train the NNs
5. **end for**

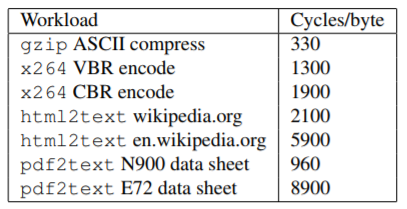
# resultados

|  |  |
| --- | --- |
| Notation | Default value |
| users | 3 |
| tasks | 3 |
|  | 1.45 × 10-7 J/bit |
|  | 1.5 × 10-7 J/bit |
|  | 100 Mbps |
|  | 10 × 109 cycle/s o 3000 cycles/bit |
|  | 3.25 × 10−7 J/bit |
|  | 4.75 × 10-7 s |
|  | 1 |

These values are taken from [41] which is based in the smartphone Nokia N810 and N900, the energy characteristics of local computing are:



In table 1, we conclude that the device has a CPU rate of 500 × 106 cycles/s and energy consumption of 730 × 106 cycles/J.

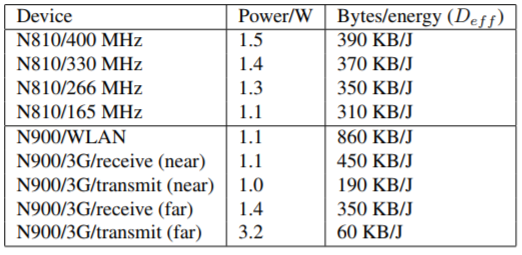


From this table we choose a video x264 CBR encoder which works at 1900 cycles/byte. With this data we can calculate and .

The local processing energy consumption per bit is:

And the local computation time per bit is:

From the next table we calculate the transmission energy consumption of the mobile device as:



<https://arxiv.org/pdf/1803.06577.pdf>

# conclusiones y líneas futuras

## Conclusiones

In this paper, we have implemented the LSTM prediction model and offloading strategy to improve the system performance. First, the LSTM model is used to predict the traffic coming from the users, aiming to provide a guideline for mobile data offloading.

After that, the offloading strategy based in a distributed deep learning algorithm, DDLO, for MEC networks is used to minimize the overall system utility including both the total energy consumption and the delay in finishing the task. The algorithm takes advantages of multiple DNNs and generates close-to-optimal solutions without manually labeled data. Numerical results have validated the accuracy of the proposed algorithm and the performance advantage compared with the existing deep Q-network algorithm. Furthermore, the proposed DDLO algorithm can generate near-optimal offloading decisions in less than one second, whose computation time is independent of the number of DNNs. We expect that such a distributed deep learning-based framework can be further extended to optimize real time offloading in future implementation of MEC networks

## Líneas futuras

# bibliografía

|  |  |
| --- | --- |
| [1] | «Portal web de la ETSIT UPM,» [En línea]. Available: http://www.etsit.upm.es/. |
| [2] | P. Taylor, Text-to-Speech Synthesis, Cambridge University Press, 2009. |
| [3] | Zen, H., Tokuda, K., & Black, A. W., «Statistical parametric speech synthesis,» *Speech Communication, 51(11),* pp. 1039-1064, 2009. |
| [4] | Pardo, J.M., Giménez de los Galanes, F.M., Vallejo, J.A., Berrojo, M.A., Montero, J.M., Enríquez, E., Romero, A., «Spanish text-to-speech, from prosody to acoustics,» de *International Congress of Acoustics*, 1995. |

1. Chen X, Jiao L, Li W, Fu X (2016) Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw 24(5):2795–2808
2. Tran TX, Pompili D (2017) Joint task offloading and resource allocation for multi-server mobile-edge computing networks. arXiv:1705.00704
3. Bi S, Zhang Y (2018) Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Trans Wirel Commun 17(6):4177–4190
4. Guo S, Xiao B, Yang Y, Yang Y (2016) Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE international conference on computer communications, pp 1–9
5. Zhang J, Xia W, Zhang Y, Zou Q, Huang B, Yan F, Shen L (2017) Joint offloading and resource allocation optimization for mobile edge computing. In: IEEE global communications conference, pp 1–6
6. Chen M, Liang B, Dong M (2016) Joint offloading decision andbandwidth allocation for multi-user multi-task mobile edge. In: 2016 IEEE international conference on communications, Kuala Lumpur, pp 1–6
7. Phaniteja S, Dewangan P, Guhan P, Sarkar A, Krishna K (2017) A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots. In: 2017 IEEE international conference on robotics and biomimetics (ROBIO), Macau, pp 1818–1823
8. Sharma A, Kaushik P (2017) Literature survey of statistical, deep and reinforcement learning in natural language processing. In: International conference on computing, communication and automation, Greater Noida, pp 350–354
9. Mnih V, Kavukcuoglu K et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
10. Dulac-Arnold G, Evans R, Hasselt H, Sunehag P, Lillicrap T, Hunt J, Mann T, Weber T, Degris T, Coppin B (2015) Deep reinforcement learning in large discrete action spaces. arXiv:1512.07679
11. Zhang C, Patras P, Haddadi H (2018) Deep learning in mobile and wireless networking: a survey. arXiv:1803.04311
12. Sun H, Chen X, Shi Q, Hong M, Fu X, Sidiropoulos ND (2017) Learning to optimize: Training deep neural networks for wireless resource management. In: Proceedings of IEEE international workshop on signal processing advances in wireless communications, pp 1–6
13. Xu Z, Wang Y, Tang J, Wang J, Gursoy MC (2017) A deep reinforcement learning based framework for power-efficient resource allocation in edge RANs. In: Proceedings of IEEE international conference on communications, pp 1–6
14. Samuel N, Diskin T, Wiesel A (2017) Deep MIMO detection. In: IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun., pp. 690–694
15. Ye H, Li GY, Juang BH (2018) Power of deep learning for channel estimation and signal detection in OFDM systems. IEEE Wireless Commun Lett 7(1):114–117
16. He Y, Zhang Z, Yu FR, Zhao N, Yin H, Leung VC, Zhang Y (2017) Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks. IEEE Trans Veh Technol 66(11):10433–10445
17. Zhong C, Gursoy MC, Velipasalar S (2018) A deep reinforcement learning-based framework for content caching. In: IEEE 52nd Annual Conference on Information Sciences and Systems (CISS),pp. 1–6
18. He Y, Yu FR, Zhao N, Leung VC, Yin H (2017) Software-defined networks with mobile edge computing and caching for smart cities: a big data deep reinforcement learning approach. IEEE Commun Mag 55(12):31–37
19. Min M, Xu D, Xiao L, Tang Y, Wu D (2017) Learning-based computation offloading for IoT devices with energy harvesting. arXiv:1712.08768
20. Chen X, Zhang H, Wu C, Mao S, Ji Y, Bennis M (2018) Performance optimization in mobile-edge computing via deep reinforcement learning. arXiv:1804.00514
21. Huang L, Feng X, Qian LP, Wu Y (2018) Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing. In: 3rd EAI International Conference on Machine Learning and Intelligent Communications, pp 1–10
22. Tan, H., Han, Z., Li, X.Y., Lau, F.C.: Online job dispatching and scheduling in edge-clouds. In: IEEE International Conference on Computer Communications (INFOCOM), pp. 1–9 (2017)
23. Li, T., Magurawalage, C.S., Wang, K., Xu, K., Yang, K., Wang, H.: On efficient offloading control in cloud radio access network with mobile edge computing. In: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 2258–2263 (2017)
24. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. 5, 2795–2808 (2016)
25. Habak, K., Ammar, M., Harras, K.A., Zegura, E.: Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In: IEEE International Conference on Cloud Computing (CLOUD), pp. 9–16 (2015)
26. Drolia, U., Guo, K., Tan, J., Gandhi, R., Narasimhan, P.: Cachier: Edge caching for recognition applications. In: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 276–286 (2017)
27. Wang, L., Jiao, L., Li, J., Muhlhauser, M.: Online resource allocation for arbitrary user mobility in distributed edge clouds. In: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 1281–1290 (2017)
28. Teerapittayanon, S., McDanel, B., Kung, H.T.: Distributed deep neural networks over the cloud, the edge and end devices. In: IEEE International Conference on Distributed Computing Systems (ICDCS), zzpp. 328–339 (2017)
29. Hao Jiang, Duo Peng, Kexin Yang, Yuanyuan Zeng, and Qimei Chen(B): Predicted Mobile Data Offloading for Mobile Edge Computing Systems , Springer Nature Switzerland AG 2018 M. Qiu (Ed.): SmartCom 2018, LNCS 11344, pp. 153–162, 2018. <https://doi.org/10.1007/978-3-030-05755-8_16>
30. Meng-Hsi Chen, Ben Liang, Min Dong: Joint Offloading and Resource Allocation for Computation and Communication in Mobile Cloud with Computing Access Point
31. N. I. Sapankevych and R. Sankar, “Time Series PredictionUsing Support Vector Machines: A Survey,” IEEEComputational Intelligence Magazine, vol. 4, no. 2,May 2009, pp. 24–38.
32. May 2009, pp. 24–38.[5] W.-C. Hong, “Application of Seasonal SVR withChaotic Immune Algorithm in Traffic Flow Forecasting,”Neural Computing and Applications, vol. 21, no. 3,Apr 2012, pp. 583–593.
33. J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, “Spatiotemporal Modeling and Prediction in Cellular Networks: A Big Data Enabled Deep Learning Approach,” Proc. 36th International Conference on Computer Communications (INFOCOM), Atlanta, GA, USA, May 2017, pp. 1–9.
34. A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human Trajectory Prediction in Crowded Spaces,” Proc. 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June 2016, pp. 961–971.
35. S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon,“Providing Public Intradomain Traffic Matrices to the Research Community,” SIGCOMM Comput. Commun.
36. A. Azzouni and G. Pujolle, “A long short-term memory recurrent neural
37. network framework for network traffic matrix prediction,” CoRR, vol.
38. abs/1705.05690, 2017.
39. [14] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014. Singapore: ISCA, 2014.
40. Hoang Duy Trinh, Lorenza Giupponi, Paolo Dini: Mobile Traffic Prediction from Raw Data Using LSTM Networks
41. A. P. Miettinen and J. K. Nurminen, “Energy efficiency of mobile clients in cloud computing,” in Proc. USENIX Conf. on Hot Topics in Cloud Comput. (HotCloud), Jun. 2010, pp. 4–11.

# anexo a: aspectos ÉTICOS, económicos, sociales y ambientales

El apartado “Requisitos de las acreditaciones internacionales EUR-ACE y ABET” de la Normativa de TFT de la ETSIT-UPM establece que “*La memoria del TFT del GITST, GIB y MUIT, y en general la de aquellas titulaciones que hayan obtenido o para las que se desee solicitar una acreditación internacional EUR-ACE o ABET, debe mostrar conciencia de la responsabilidad de la aplicación práctica de la ingeniería, el impacto social y ambiental, el compromiso con la ética profesional, la responsabilidad y las normas de la aplicación práctica de la ingeniería, así como sobre las prácticas de gestión de proyectos, gestión, y control de riesgos, entendiendo sus limitaciones*”.

Este anexo obligatorio del TFT tendrá un carácter sintético con los siguientes apartados:

## A.1 iNTRODUCCIÓN

Breve descripción del contexto del proyecto, objetivos, necesidades que pretende cubrir o problemas que pretende resolver, centrándose en su relación con los temas sociales, económicos, éticos, legales y/o ambientales que se hayan identificado.

## A.2 DESCRIPCIÓN DE IMPACTOS RELEVANTES RELACIONADOS CON EL PROYECTO

Síntesis del trabajo realizado en la fase 2, de selección y descripción de impactos. Presentar y justificar las conclusiones a las que se haya llegado sobre cuáles son los asuntos más relevantes relacionados con la sostenibilidad social, económica o ambiental, así como los principales grupos de interés identificados y que se han considerado en los análisis posteriores.

## A.3 ANÁLISIS DETALLADO DE ALGUNO DE LOS PRINCIPALES IMPACTOS

Síntesis del trabajo de análisis realizado.

## A.4 CONCLUSIONES

Valorar el proyecto desde un punto de vista ético, social, económico y medioambiental y justificar si el uso de criterios de sostenibilidad ha aportado o puede aportar valor añadido al proyecto.

# anexo b: presupuesto económico

El apartado “Requisitos de las acreditaciones internacionales EUR-ACE y ABET” de la Normativa de TFT de la ETSIT-UPM establece que “*La memoria del TFT del GITST, GIB y MUIT, y en general la de aquellas titulaciones que hayan obtenido o para las que se desee solicitar una acreditación internacional EUR-ACE o ABET, … debe incluir un presupuesto económico*”. A modo de ejemplo, la tabla del presupuesto de un proyecto podría ser la siguiente:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COSTE DE MANO DE OBRA (coste directo)** | | | | | **Horas** | | **Precio/hora** | **Total** |
|  | | | | | 300 | | 15 € | **4.500 €** |
|  | | | |  |  | |  |  |
| **COSTE DE RECURSOS MATERIALES (coste directo)** | | | | **Precio de compra** | **Uso en meses** | | **Amortización (en años)** | **Total** |
| Ordenador personal (Software incluido)....... | | | | 1.500,00 € | 6 | | 5 | 150,00 € |
| Impresora láser | | | | 500,00 € | 6 | | 5 | 50,00 € |
| Otro equipamiento | | | |  |  | |  |  |
|  | | | |  |  | |  |  |
| **COSTE TOTAL DE RECURSOS MATERIALES** | | | | | | | | **200,00 €** |
|  | |  | |  | |  | |  |
| **GASTOS GENERALES (costes indirectos)** | | 15% | | sobre CD | | | | **705,00 €** |
| **BENEFICIO INDUSTRIAL** | | 6% | | sobre CD+CI | | | | **324,30 €** |
|  | |  | |  | |  | |  |
| **MATERIAL FUNGIBLE** | |  | |  | |  | |  |
| Impresión | | | | | | | | **100,00 €** |
| Encuadernación | | | | | | | | **300,00 €** |
|  |  | |  | | |  | |  |
| **SUBTOTAL PRESUPUESTO** | | | | | | | | **6.129,30 €** |
| **IVA APLICABLE** | | | | | | | 21% | **1.287,15 €** |
|  |  | |  | | | |  |  |
| **TOTAL PRESUPUESTO** | | | | | | | | **7.416,45 €** |

Esta tabla podría ser rellenada mediante una sencilla hoja de cálculo como la siguiente:

