



**Innovative algorithms for completion of
resource intensive tasks in IoT devices
and novel applications in the Smart City
& Smart Building**

**Author:
Asier Garmendia Orbegozo**

Advisors:

Associate Prof. Jose David Nuñez-Gonzalez
PhD Miguel Angel Anton Gonzalez

2024

Acknowledgment

I would like to begin by thanking my thesis director, José David Núñez-González. This thesis would not be possible without the uninterrupted and unconditional availability throughout the entire period of my stay at the university. Both in academic matters and in more practical matters it has been a fundamental pillar in the years that the thesis has lasted. Likewise, I would like to thank the availability and help provided by Miguel Ángel Antón, co-director of this thesis at TecNALIA, who has offered us the opportunity to use powerful computational resources in addition to providing valuable knowledge in many aspects.

Likewise, I appreciate all the support that my parents have given me, who have been of special importance for the development of the thesis. I was encouraged to start this doctoral career when I was considering other options along with the possibility of obtaining the doctorate.

I would also like to thank Itziar Alonso from TecNALIA who opened the possibility for me to do the doctorate when I was doing my master's thesis at the research center.

Reconocimiento

Me gustaría comenzar agradeciendo a mi director de tesis, José David Núñez-González. Esta tesis no sería posible realizarla sin la disponibilidad ininterrumpida e incondicional durante todo el período de mi estancia en la universidad. Tanto en cuestiones académicas como en cuestiones más prácticas ha sido un pilar fundamental en los años que ha durado la tesis. Asimismo, me gustaría agradecer la disponibilidad y ayuda brindada por Miguel Ángel Antón, codirector de esta tesis de Tecnia, quien nos ha ofrecido la oportunidad de utilizar potentes recursos computacionales además de aportar valiosos conocimientos en muchos aspectos.

De igual manera agradezco todo el apoyo que me han brindado mis padres, quienes han sido de especial importancia para el desarrollo de la tesis. Me animaron a iniciar esta carrera de doctorado cuando estaba considerando otras opciones junto con la posibilidad de obtener el doctorado.

Me gustaría agradecer también a Itziar Alonso de Tecnia que me abrió la posibilidad de realizar el doctorado cuando estaba realizando el trabajo fin de máster en el centro de investigación.

Aitortza

Hasteko, nire tesi zuzendaria izan den José David Nuñez-González eskertu nahi nuke. Tesi hau ezingo nuke burutu ez bazen bere etengabeko prestutasuna bitarteko tesiak iraun duen urteotan. Arlo akademikoan zein kontu praktikoa-goetan ezinbesteko zutabea izan da tesiak iraun duen urteotan niretzat. Era berean, Miguel Angel Antonek eskainitako prestutasuna eskertu nahi nuke, berau zuzendariordea izanik tesi honetan. Tecnaliako partaide gisa, bertako baliabide konputazional aurreratuak eskuragarri jartzeaz gain ezagutza baliotsuak eskaini baitizkigu hainbat arlotan.

Modu berean, nire gurasoek eskaini didaten babesa eskertu nahiko nuke, oso garrantzitsuak izan baitira tesiaren garapenerako. Tesiaren ibilbidea has nezan animatu ninduten beste aukera batzuen artean nuenean tesiarekin hastea.

Azkenik, Tecnaliako Itziar Alonsori eskerrak eman nahi nizkioke, doktoretza tesia burutzeko aukera ireki zidalako, ikerketa zentro berean master amaierako lana burutzen ari nintzen garaian.

Resumen

Con la irrupción de los dispositivos electrónicos al alcance de la ciudadanía en los últimos años, estos ofrecen distintas posibilidades en cuanto a obtención y manipulación de la información y generación de nuevo conocimiento se refiere. Debido a los grandes avances tecnológicos, cada vez se pueden encontrar dispositivos con menor tamaño y consumo con mayores prestaciones computacionales capaces de ofrecer distintos servicios a los usuarios, favoreciendo y facilitando la vida de estos. Los dispositivos llamados IoT (Internet Of Things) por sus siglas en inglés, ofrecen al usuario la posibilidad de medir distintos parámetros del entorno, como intercomunicar entre ellos como con otros computadores más potentes. En una área de red local, se pueden encontrar además de estos dispositivos reducidos, otros llamados computadores Edge, que permiten la unión de estos dispositivos con otros servidores remotos, además de ofrecer otros servicios como la interconexión entre distintos dispositivos. Los dispositivos IoT se podrían clasificar en una área de red local en la capa IoT, los computadores formarían la capa Edge, y finalmente, en la capa de la nube o Cloud se encontrarían los servidores remotos de la nube.

Cada vez son más las aplicaciones que requieren de una acción inmediata en la capa del usuario final o IoT, que al estar formado por dispositivos IoT, tiene unas características determinadas. Estos dispositivos al ser de tamaño reducido, para facilitar su transporte, versatilidad a la hora del despliegue y reducir los costes de energía para alargar su funcionalidad sin necesidad de ninguna intervención, no están dotados de procesadores tan potentes como otros computadores de otras capas. Del mismo modo, las memorias de estos dispositivos no son del mismo tamaño que los que se pueden encontrar en otros dispositivos como los computadores Edge, que ofrecen suficiente memoria para abordar una gran variedad de cómputos relacionados con la ejecución de distintas tareas. Por ello, es preciso hacer modificaciones en las tareas o en las distribuciones en la carga de trabajo entre distintos dispositivos para hacer viable la ejecución de tareas complejas en el menor tiempo posible.

La ejecución cerca de los dispositivos IoT o dispositivos de los usuarios finales, es esencial para garantizar que la ejecución de tareas críticas con el tiempo de respuesta se completen en el menor tiempo posible, ya que la transmisión de la información referida a dicha tarea por la red conlleva un importante retraso

que dificulta el cumplimiento del requerimiento de la latencia. Aun siendo la capa que ofrece las mayores prestaciones, la capa de la nube se convierte el peor destino de computación en estos entornos, al no poder cumplir con dichos requerimientos. En cambio, los computadores Edge están suficientemente capacitados para llevar a cabo tareas que requieren de cálculos computacionales de gran complejidad y gasto, y lo suficientemente cerca de los dispositivos finales para que la transmisión de la información entre dispositivos no sea ningún inconveniente.

Para dar una respuesta optimizada en varias aplicaciones es preciso el uso de técnicas de Machine Learning o Aprendizaje Automático, que usan arquitecturas o modelos complejos que muchas veces no es viable su entrenamiento e implementación en dispositivos reducidos. En los últimos años, han sido muy relevantes en distintos campos las técnicas de Aprendizaje Profundo o Deep Learning, siendo las redes neuronales su principal herramienta. Estas redes, al estar compuestas por distintas capas y un sinnúmero de parámetros para su representación requieren mucha memoria y capacidad de cómputo para su entrenamiento que no les pueden ofrecer los dispositivos reducidos en versión original. Aun entrenando en otros computadores más potentes e implantando una vez entrenados para su fase de inferencia en los dispositivos finales, es difícil que estos dispositivos alberguen de suficiente memoria para su representación. Es por ello, que en esta tesis uno de los objetivos principales ha sido la reducción de estas redes a través de diferentes técnicas. No obstante, los modelos reducidos es posible que no cumplan con los requisitos de precisión establecidos por las aplicaciones, debido a que al reducir sus dimensiones pasa lo mismo con su capacidad de predicción.

Otra opción muy interesante para posibilitar la ejecución de tareas complejas con una latencia baja y con suficiente precisión es la de enviar las tareas desde dispositivos finales a dispositivos de la capa Edge, al estar estos últimos dotados de mayores prestaciones para su entrenamiento como ejecución. De este modo, no habría necesidad de reducir los algoritmos empleados para dar la respuesta en cada tarea y el resultado final conservaría la capacidad de predicción de los modelos originales. Uno de los problemas al emplear esta técnica de descarga de tareas es la posible sobrecarga de varios computadores al estar recibiendo las tareas de una multitud de dispositivos finales. Para evitar este tipo de conflictos es necesario el empleo de una política de descarga que permita la distribución equitativa de tareas entre los computadores cercanos de la capa Edge con el fin de evitar las citadas sobrecargas en estos dispositivos y la ejecución de todas las tareas en la latencia requerida.

Finalmente, durante la tesis como tercer objetivo se ha establecido la elaboración de una aplicación innovativa que utilizara las mediciones obtenidas por los dispositivos IoT que al fusionar con otras fuentes de información ofreciera un conocimiento novedoso. En este caso, se ha optado por utilizar las mediciones obtenidas por los sensores colocados en las puertas de distintos tipos de edifi-

cios en ciudades de Madrid y Melbourne. Utilizando técnicas de aprendizaje no supervisado, que se centra en encontrar relaciones entre datos sin que estos estén etiquetados, se ha pretendido identificar los distintos edificios partiendo de las mediciones de los sensores que interpretaban la actividad de los ciudadanos cerca de estos edificios.

La metodología empleada en el transcurso de la tesis para llegar a cumplir los objetivos preestablecidos con sendos trabajos que finalmente han sido publicados en revistas científicas de impacto, ha sido similar en todos los casos. Primero, se ha detectado una deficiencia o flaqueza en la literatura. Esta podría ser una falta de técnica o conocimiento concreto o una posibilidad de mejora en una aplicación o tecnología concreta. Acto siguiente, se ha desarrollado una solución optimizada que mejorará los resultados obtenidos en otros trabajos o se ha desarrollado una técnica novedosa que ofreciera una visión diferente o conocimiento innovador. Para dar firmeza a las soluciones propuestas, estas se han contrastado utilizando bases de datos abiertos con las alternativas que mejores resultados ofrecían en la literatura.

En el primer trabajo, el objetivo principal fue la reducción de los modelos de Aprendizaje Profundo, que al fin y al cabo se ha centrado en reducir las dimensiones de las redes neuronales para que puedan ajustarse a las restricciones de memoria establecidas por los dispositivos finales. Para llegar a dicho objetivo existen distintas posibilidades entre las que destacan técnicas de podaje o pruning y técnicas de cuantización. La primera variante se centra en reducir las conexiones o neuronas de las diferentes capas que forman las redes neuronales. Para ello, se establece una política de relevancia para establecer un orden entre distintos componentes que prioriza la eliminación de ciertos componentes por encima de otros. Finalmente, se eliminan los componentes menos relevantes en el resultado final, acelerando la capacidad de respuesta del modelo resultante y reduciendo su tamaño. En el segundo, los pesos que representan las conexiones entre neuronas al estar representados en formato de cierta cantidad de bits, estos pueden reducir el tamaño que ocupan en la memoria al utilizar un número de bits menor. Esto se consigue al reducir el número de bits utilizado para su representación con la consiguiente reducción de precisión del modelo.

Al ver que en la literatura el uso de técnicas de podaje era el predominante con el objetivo de preservar la capacidad de predicción del modelo resultante y reducir la memoria que ocupa en el dispositivo final, optamos por mejorar alguna variante que ofreciera resultados subóptimos en términos de eficiencia y precisión. La descomposición en valores singulares (SVD), ha ofrecido una reducción en la memoria significativa al reducir la dimensión de la matriz representante de una capa formante de una red neuronal en otras tres matrices de considerable menor tamaño. En estas matrices los valores de mayor relevancia se colocan en las primeras filas y columnas y conservando solo estas, se consigue una mayor reducción de tamaño del modelo resultante. No obstante, la relevancia de cada componente puede ser determinado por distintos criterios, que en

este caso el que mejor resultado ofrece es el coste. Este último es la diferencia que ofrece el modelo resultante en cuanto a la precisión se refiere cuando al modelo se le elimina dicho componente comparado con la precisión del modelo original. Así, los elementos de mayor coste son los que preservan en el modelo final.

A pesar de ser una técnica con un rendimiento alto en varias bases de datos abiertas, es verdad que las capas de las redes neuronales no son independientes entre sí, y que la técnica previamente mencionada no tiene en cuenta la relación existente entre estas capas. Por ello, en el primer trabajo que constituye esta tesis se utiliza la idea de que las relevancias de los componentes de distintas capas están relacionadas por las interconexiones entre ellas. Es decir, la relevancia de un componente de una capa no viene dada por su relevancia individual solamente, sino que también influyen las conexiones con los componentes de otras capas subyacentes y las relevancias de los componentes de estas capas. Las relevancias de los componentes se propagan hacia atrás empezando por la capa final de una red neuronal a través de las conexiones con los componentes de otras capas.

Para verificar nuestra hipótesis se realizó una comparación con otro trabajo de la literatura que utilizaba una optimización de la descomposición de los valores singulares de las matrices representantes de una capa de la red en varias bases de datos abiertas. Se vio que en ciertos casos nuestra alternativa ofrecía un resultado mejor pero que en otros la versión de la literatura ofrecía un mejor rendimiento.

En el segundo trabajo, el objetivo principal ha sido el establecimiento de una técnica de distribución de tareas entre computadores constituyentes de la capa Edge e IoT en una área de red local. Al no llegar a la precisión pretendida por las aplicaciones utilizando técnicas de reducción de los modelos utilizados en el primer objetivo, la distribución de tareas entre distintos computadores con mayores prestaciones que los dispositivos finales es una alternativa eficaz que ofrece un tiempo de respuesta aceptable sin reducir los modelos utilizados para la ejecución de las tareas. Aunque la solución propuesta en este segundo trabajo en un principio puede dar la impresión de que no puede tener ningún inconveniente, es verdad que en determinadas situaciones en las que los dispositivos finales envían varias tareas a la vez a estos computadores Edge, estos últimos pueden sobrecargarse y no ofrecer la respuesta requerida por los dispositivos finales en sus respectivas tareas. Para evitar estos conflictos es preciso el empleo de una política de descarga de tareas que garantice la ejecución de todas las tareas en la latencia requerida, a la vez que evite la sobrecarga de determinados computadores y evite la congestión de tráfico de la red.

Los grafos, compuestos por nodos y arcos que unen estos nodos se asemejan mucho a la arquitectura de una área de red local. Debido a esta similitud, en el segundo trabajo se optó por utilizar las redes neuronales basadas en grafos para

determinar en cada situación el destino óptimo de descarga de una tarea dentro del área de red local. Los nodos representaban los computadores y los arcos las interconexiones entre estos. Las características de los nodos venían determinados por las características computacionales (capacidad de procesamiento, memoria, etc.) y por las características de las tareas (longitud de la tarea y latencia), y las características de los arcos por la valoración del arco, que es la relación entre las tareas ejecutadas satisfactoriamente utilizando la conexión como vía de descarga, y todas las tareas descargadas utilizando dicha conexión.

Sin embargo, el aprendizaje por refuerzo o Reinforcement Learning ofrece la posibilidad de considerar los últimos cambios del entorno a analizar con la consiguiente mejor interpretación de este. Por ello, se utilizaron en paralelo técnicas de Deep Q-Network que aplica el uso de redes neuronales en el ámbito del aprendizaje por refuerzo. El aprendizaje por refuerzo se basa en un agente y su entorno, en el que el agente a través de su política de decisión determina la acción óptima a tomar en cada instante y recibe del entorno el siguiente instante y la recompensa por la acción tomada. La optimización se centra en maximizar la recompensa acumulada a través de las consecutivas acciones tomadas en el entorno. En esta casuística, las posibles acciones son los posibles destinos de descarga de las tareas, la recompensa positiva en caso de cumplimiento de la tarea en la latencia requerida y negativa en caso contrario, y las características que determinan el entorno serían las características de los computadores, las características de las tareas y el estado de los computadores y la red.

Las dos alternativas propuestas se compararon utilizando un simulador de computación en Edge llamado PureEdgeSim, frente a las políticas de descarga que tiene por defecto el simulador. Se compararon el rendimiento obtenido por las distintas técnicas y la distribución de tareas entre distintas capas, entre otras métricas. Se vio que en casos en los que entre los dispositivos finales se encontraban los computadores Edge se obtenían los mejores resultados. Al contrario, cuando el destinatario era un servidor remoto de la nube, los resultados estaban lejos de ser los óptimos debido a que en varias tareas el requerimiento de la latencia no se cumplía por el atraso acarreado al atravesar la red entre la conexión entre el dispositivo final y el servidor. Al aumentar los dispositivos finales que requerían el uso de otros computadores más potentes para la ejecución de tareas asignadas a ellas, se veía un aumento de tareas ejecutadas entre los dispositivos finales al no disponer de suficientes computadores Edge para tantas tareas. En estos casos, las políticas de descarga propuestas eran capaces de distribuir las tareas más complejas entre los computadores Edge y las más simples entre los dispositivos IoT. Ambas alternativas propuestas mejoraban el rendimiento ofrecido por las políticas por defecto del simulador, la basada en el aprendizaje por refuerzo siendo superior por su capacidad de consideración de los últimos cambios en el entorno.

En el último trabajo que constituye la tesis, se utilizaron técnicas de agrupamiento y redes neuronales basados en grafos para ofrecer un nuevo conocimiento

en el ámbito del Smart City partiendo de simples métricas de actividad de los ciudadanos en los alrededores de diferentes tipos de edificios. Los sensores colocados en frente de los edificios contabilizaban los pasos de los peatones, que con un preprocesamiento de los datos previo se obtuvieron medias de estos durante distintas franjas horarias. Utilizando este tipo de información se consigue clasificar los distintos tipos de edificios basándose en la actividad de los ciudadanos en distintas franjas horarias de distintos tipos de días (semanales o de fines de semana) y la tipología de los edificios circundantes.

Al disponer de una gran cantidad de características por sensor, al disponer de medias en franjas horarias de dos horas por dos tipos de días, lo preciso es reducir la dimensionalidad de la problemática utilizando diversas técnicas. En este trabajo se optó por utilizar la técnica de Análisis de Componentes Principales (PCA) para reducir las características que representan a cada sensor. Acto siguiente, se llevaron a cabo dos agrupamientos basándose en dos criterios. El primero se basó en las medias de las activaciones de estos sensores en las franjas horarias de los diferentes días, y el segundo se basó en el número de edificios circundantes en un radio de 300 m desde cada sensor.

Con la finalidad de optimizar las agrupaciones obtenidas en el proceso de agrupamiento, se empleó el entrenamiento de la red basada en el grafo para obtener un agrupamiento más diferenciador que en los procesos previos. Los nodos constituyen los sensores, las características de los nodos las activaciones medias de los sensores en las distintas franjas horarias, y las características de los arcos las inversas de las distancias entre sensores, para incluir el componente de la tipología de los edificios circundantes.

Para verificar la bondad de los agrupamientos, se emplearon modelos de aprendizaje supervisado atribuyendo como etiqueta el grupo asignado a cada sensor en el proceso de agrupamiento. Atendiendo a la precisión obtenida por el modelo, se verificó la bondad de los agrupamientos, viendo que el proceso que atendía a la tipología de los edificios circundantes era superior al otro proceso de agrupamiento. Del mismo modo, la validación del grafo se llevó a cabo dividiendo los sensores de cada grupo en ejemplos de entrenamiento y testeo, obteniendo las mismas métricas de rendimiento que en el proceso de validación anterior. Se vio en los resultados que el agrupamiento basado en la tipología de los edificios circundantes era superior al resto, también a la alternativa que hacía uso de las redes neuronales basadas en los grafos, este último ofreciendo una ligera mejoría respecto al agrupamiento basado en la actividad de los peatones.

Como conclusión, en el primer trabajo se vio que la relevancia de los componentes entre distintas capas podría tener su impacto en el resultado final a la hora de determinar los elementos menos representativos para el podaje de una red neuronal. A su vez, ambas técnicas contrapuestas eran costosas en tiempo y recursos computacionales, y el criterio que menos tiempo y recursos requería para determinar los elementos a podar podría basarse en solamente atender la mag-

nitud de los pesos de las matrices. En el segundo trabajo, aunque los resultados eran satisfactorios, no se podrían comparar directamente con otros trabajos de la literatura al disponer de entornos totalmente diferentes y experimentos distintos. A su vez, sería preciso contar también con el consumo energético de los computadores como característica diferencial a la hora de establecer la política de descarga entre distintos puntos de computación. En el tercer trabajo, al disponer de una información novedosa y relevante en el ámbito de una ciudad inteligente, se podría emplear el nuevo conocimiento generado como punto de partida para una óptima distribución de energía y alumbrado entre distintos tipos de edificios.

Abstract

This thesis has been focused on the topic of Machine Learning, where we have been working on the refinement of different methods from the literature, and diverse applications related to Smart Cities and Edge Computing. Precisely, the main contributions have been made by improving algorithms to ease their computation in resource constrained devices, establishing policies for orchestrating load distribution between these devices and the acquisition of new knowledge based on the data obtained by these devices through long periods of time, opening the way to novel applications.

The first work focuses on the refinement of methods for reducing the Neural Networks on resource constrained devices. Due to the lightweight nature of these devices it is not viable to carry out computation of these algorithms on them, so that, variants for reducing or modifying these algorithms have been proposed. The principle of backpropagation of the relevances of components of the network has been imposed to a relevant work from the literature to offer a new way for shrinking Neural Networks.

The second contribution is related to the paradigm of offloading of tasks in Edge Computing between different computation centers. The offloading policy that ensures load balance and computation of all tasks within the required latency is pursued. We have implemented a Graph Neural Network for reproducing the local network structure, including the devices' properties and network state as features of the graph. In parallel, we used the RL technique Deep Q-Network due to the constant updates of the network environment, and the ability of RL algorithms to learn these constant changes of the networking environment. We tested both task offloading policies in a well-known simulator, and we outperformed the default policies of the simulator.

The last work consists of a novel application for Smart Cities based on the readings of sensors located outside different buildings in a city. These data, collected over several months, was a source of new knowledge about the essence of these buildings. Sensor reading patterns reflect the pedestrians activity in these buildings. Consequently, a classification of buildings was achieved based on the typology of buildings surrounding these sensors and the pedestrians' mobility pattern at different time intervals of different days of the week.

Contents

1	Synthesis	1
1.1	Introduction	1
1.2	Theoretical framework	7
1.3	Objectives & Research Methodology	15
1.3.1	General objectives	15
1.3.2	Research methodology	15
1.3.3	Specific objectives	16
1.4	Discussion of results	31
1.4.1	Neural Network Reduction	31
1.4.2	Offloading of tasks in the Edge	32
1.4.3	Cooperation of Edge devices	35
1.5	Relevance of Results	38
2	Conclusions & Discussion	40
A	Articles	49
A.1	SLRProp: A Back-Propagation Variant of Sparse Low Rank Method for DNNs Reduction	49
A.2	Task Offloading in Edge Computing using GNNs and DQN	64
A.3	Graph Based Learning for Building Prediction in Smart Cities	88

List of Acronyms

AI	Artificial Intelligence
AUC	Area Under the ROC curve
BOSME	Bayesian network-based over sampling method
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DL	Deep Learning
DNN	Deep Neural Networks
DQN	Deep Q-Network
DT	Decision Tree
EC	Edge Computing
FC	Fully Connected
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
FRL	Final Response Layer
GB	Gyga Byte
GCN	Graph Convolutional Neural Networks
GNN	Graph Neural Networks
HAIS	Hybrid Artificial Intelligence Systems
HCD	Highest Computing Device
HDR	Highest Data Rate Offloading
ICMSLE	International Conference on Statistics and Machine Learning
IEEE	the Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ITS	Intelligent Transportation Systems
JIF	Journal Impact Factor
MDPI	Multidisciplinary Digital Publishing Institute
MIP	Millions of Instructions per Second
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Networks
PCA	Principal Component Analysis
RAM	Random Access Memory
RF	Random Forest

RL	Reinforcement Learning
RT	Real-Time
SVD	Single Value Decomposition
SLR	Sparse Low-Rank Method
SVM	Supported Vector Machine
TB	Tera Byte
TN	True Negatives
TP	True Positives
TPR	True Positive Rate
Wh	Watt-hour

List of Figures

1	Fields of use of IoT devices	2
2	Overview of a Smart City	3
3	Subcategories of Machine Learning.	5
4	Different layers of a network	8
5	Graphical representation of a Neural Network	10
6	Single Value Decomposition	18
7	Overview of the training procedure of DQN.	24
8	Overview of the entire process.	26
9	Cumulative explained variance for the Melbourne building fea- ture based case and pedestrians with mean hourly mean count feature based case.	29
10	Task distribution with different algorithms and number of Edge devices	34
11	Success rates with different algorithms and number of Edge devices	35
12	Clustering based on the typology of buildings. Black points indi- cate the centroids of each cluster.	36

List of Tables

1	SLR vs. SLRProp accuracies for different datasets.	32
2	Success rate of different algorithms including different type of destiny devices (10 Edge devices).	33
3	Success rate of different algorithms including different type of destiny devices (20 Edge devices).	33
4	Success rate of different algorithms including different type of destiny devices (30 Edge devices).	34
5	Performance metrics for Building typology dataset based clustering.	37
6	Performance metrics for time-interval mean activation dataset based clustering.	38
7	Comparative of results with the literature.	39

Chapter 1

Synthesis

In this synthesis Chapter an overview of this thesis is given to ease the understanding of it. The subject of the thesis is presented in this Section and theoretical insights are given as well. General objectives are defined and the results obtained in the works developed during the thesis are summarized. The Chapter is organized as follows: Section 1.1 introduces the readers to the topic of research. Section 1.2 gives a more detailed overview of the theoretical framework. In Section 1.3 general objectives of research are described and the results obtained through the investigation are summarized in Section 1.4.

The rest of the thesis follows this division: Chapter 2 introduces the conclusions made after the entire investigation have been carried out. Finally, in the Appendix Chapter the published papers of the works that make up the thesis are attached.

1.1 Introduction

During the last decades the use of **Internet of Things (IoT)** devices has drastically increased, in such a way that we can nowadays find in any scenario at any moment tens of devices of these nature surrounding us. IoT devices are atypical computing devices that have the ability to connect wirelessly to a network and transmit data. These are intended to respond to different events of the environment where they are located, sense different parameters or collect data related to those events or parameters. Examples of such devices could be found in a **Smart Home** (different devices interact with each other to offer conveniences to the residents), in a Smart Building (smart sensors could give an updated state of all available halls) or in a Smart Factory (sensors could provide information about anomalies in an assembly plant). In the same way, there are other type of devices called **Edge Datacenters** in a local networking area such as gateways, which offer different services like communication with a cloud remote server, routing of network traffic or aggregation and preprocessing

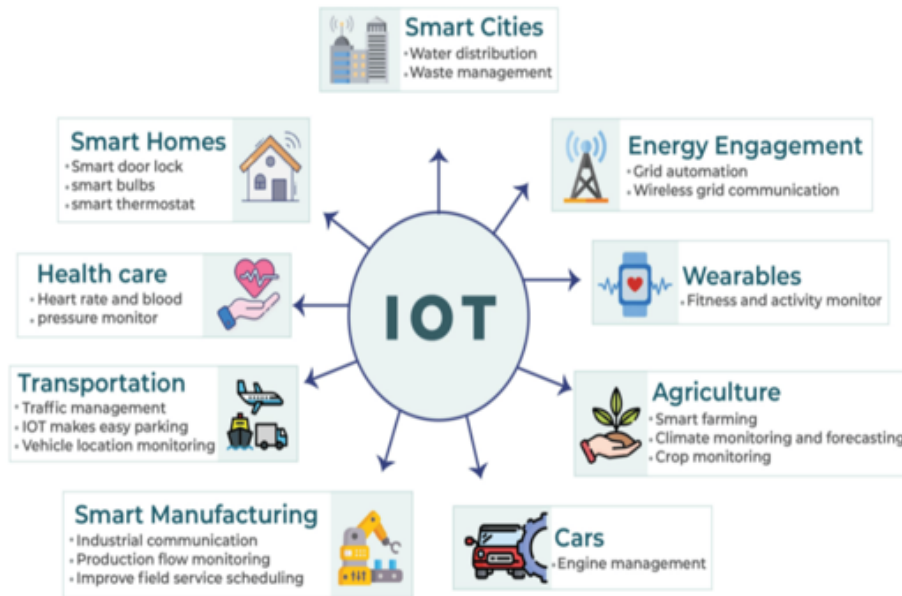


Figure 1: Fields of use of IoT devices

Source: Alghofaili, Y.; Rassam, M.A. A Trust Management Model for IoT Devices and Services Based on the Multi-Criteria Decision-Making Approach and Deep Long Short-Term Memory Technique. *Sensors* 2022, 22, 634. <https://doi.org/10.3390/s22020634>

of data among others. These devices reduce latency and optimize bandwidth compared to the cloud servers, becoming an interesting alternative for different applications. In this PhD thesis paradigms related to computation of tasks in resource constrained devices have been studied in depth.

Due to the reduced size of these IoT devices, it is non-viable to provide them with long memory and processors with high capabilities, among other resources. The nature of these devices requires minimal resources to preserve their lightweight essence, but being capable to respond immediately to different events. Consequently, in the majority of the cases these devices need to offload their tasks to more powerful computers that are provided with sufficient memory and high-powered processors. However, there are some cases in which it could be interesting to carry out more complex calculations without sending them to a remote server. Modifying the algorithms used to solve the tasks assigned to these tiny devices is one of the alternatives to achieve it.

Instead, in other scenarios where there are powerful Datacenters close to the IoT devices that can alleviate the workload on these tiny devices, the offloading of tasks from source devices to more resource rich computers offers multiple

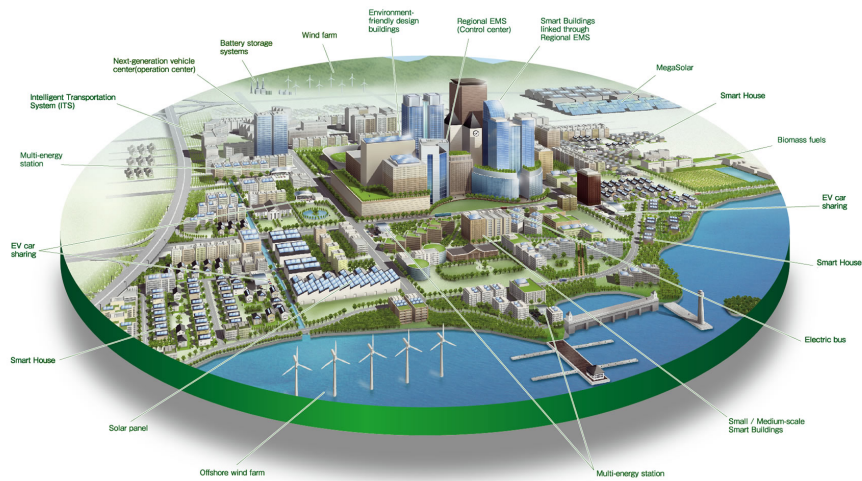


Figure 2: Overview of a Smart City

Source: <https://www.mercadoit.com/blog/noticias-it/smart-city-sostenibilidad-para-todos/>

benefits. Task offloading can be defined as the transfer of resource-intensive computational tasks to an external, resource-rich platform such as the ones used in Cloud or Edge[1]. One of them, is the possibility to compute complex models without the need to adjust these architectures. However, establishment of an optimized orchestration between different devices is essential to alleviate the workload on them. Otherwise, some Datacenters would collect too many tasks facing the overloading issue.

Smart Cities and **Smart Buildings** are becoming increasingly more popular terms in the society. These are entities that use technology to enable efficient and economical use of resources. Finally, possible potential applications are described and analyzed. Collection of data through long periods of time could be a potential source of information for obtaining valuable and novel knowledge in Smart Cities and Smart Buildings.

The use of IoT devices is increasing exponentially in different areas such as Smart Cities, Smart Home or Healthcare among others. The optimization of their use is a fundamental issue as an enhanced performance would benefit users and the environment. From a wide range of conveniences that metropolitan areas could obtain from these applications to the breakthroughs that have been making in medicine, there are a multitude of benefits that developing of different algorithms to adequate into the application and the environment in which they are supposed to be run brings with it.

Different applications that are assigned to these type of devices solicit the use of statistical methods and algorithms. To this end, **Artificial Intelligence (AI)** is the way to achieve the desired solution on these applications. In the summer of 1956 at the Dartmouth College convention, John McCarthy defined the AI like "the science and engineering to make intelligent machines". Machine Learning (ML) was launched in 1959 [2], being a subcategory of AI in the field of Computer Science that makes use of Mathematics and Statistics, to give computers the ability to learn. Thus, ML is a subarea of the AI that connects Computer Science with the Mathematics and Statistics. Within ML there are different subcategories that are the following:

- **Supervised Learning** learns a function that connects input data to output data based on training input-output instances. Its main purpose is the task of prediction of one or various output variables starting from the input variables, using a model that learns a relationship between instances whose inputs and outputs are known. Different types of Supervised Learning algorithms could be found that are divided in two main groups. The **classification** algorithms cover all the algorithms whose output values are limited within a set of values. The **regression** algorithms cover all the algorithms that the outputs can have continuous values within a range.
- **Unsupervised Learning** covers all the algorithms that find the structure or the relationships between different variables of a dataset, finding relationships and dividing the instances into clusters. In this case, the instances are not labelled.
- **Reinforcement Learning (RL)** is the subcategory that concerns with how agents should take actions in an environment with the objective of maximizing a cumulative reward. The two main elements are the environment, which is the representation of the problem to be solved including all the components that are involved in the paradigm in question, and the agent, the algorithm that is going to be learned.

Among the different possibilities of Supervised Learning, and more specifically classification algorithms, Deep Learning (DL) techniques are of special interest where **Neural Networks (NNs)** are its main algorithm. NNs trains computers to process data in a similar way that human brain does. The use of **Deep Neural Networks (DNNs)**, which are feedforward network with multiple hidden layers, in different environments and other architectures related to ML has emerged in such a way that currently NN designs have billions of parameters with a great capability of prediction, being one of the most used type of architecture in prediction tasks. Among others, some of those applications include image, sound, and textual data recognition. In contrast to other ML algorithms, the DNNs have achieved a remarkable accuracy in many scenarios. However, the use of these networks in memory and processing resource constrained devices becomes complicated due to the amount of data needed to develop these architectures and the high computation costs for training them.

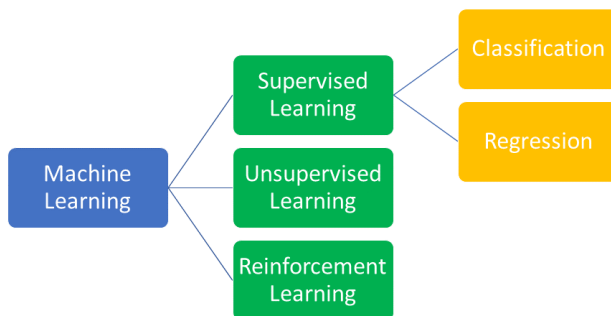


Figure 3: Subcategories of Machine Learning.

In many cases, there are different alternatives to tackle these problems, starting from sending tasks' information to other powerful computation centers to modifying the tasks themselves or the tools required to solve the tasks, as mentioned above.

Within the Supervised Learning methods as well there are other approaches very interesting like **Graph Neural Networks (GNNs)**. A Graph is composed of a set of nodes(V) and edges(E), represented by $G(E, V)$, whose data structure is non-linear. The edges connect any two nodes in the graph. Any system consisting of space and time structural relationship information can be contemplated as a space-time graph. NNs can deal with static structures and time-varying features of the graph.

$$G = (V, E, X_v(t), X_e(t))$$

These NNs are trained to carry out time series analysis using the time-varying characteristics of the graph. In the area of ML time series analysis, time series data sets are different from spatial data because they are causally consistent, meaning that data from the past is highly correlated with data from the present and future. A Graph Convolutional Network (GCN) differs from GNN that the NN that handles its features is a Convolutional Neural Network (CNN). The operation of convolution multiplies the input neurons with a set of weights that are known as filters or kernels. The filters across the whole input and enable CNNs to learn features also from neighboring cells acting as a sliding window. Within the same layer, the same filter will be used throughout image, this is referred to as weight sharing[3].

The objective of this thesis is to focus on the solutions to the computational incapacities [4] faced by these devices that could be different IoT devices, such as sensors, actuators or small devices among many others. At the same time, the collaboration of IoT devices could be beneficial for different reasons, like reduction of latency [5], computational costs, enhancement of performance or possibility to offer new services that would not be possible separately[6].

Furthermore, the cooperation of these devices would be a great resource from a wider perspective. The combination of data collected by these devices could be a source of new knowledge offering new opportunities to different type of entities. For instance, in a Smart Building or a Smart City, Real-Time (RT) measurements from sensors could give to the users information about the actual traffic situation or agglomerations of citizens. Therefore, people could avoid different traffic routes or overcrowded streets. In a Smart Building the cooperation of sensors and actuators make the lifestyle of residents convenient by providing an optimal use of household appliances and autonomous regulation of these, achieving the desired conditions previously imposed.

For these reasons, this work extends the literature giving new alternatives to cover the benefits mentioned above. Some of them are focused on reducing the dimension of the architectures [4] used for offering the service themselves without relegating different tasks to other devices and others on reorganizing the load distribution between different computation centers in a local networking area [5]. Different strategies have been proposed and many other applications are studied. The use of these techniques reduces the time needed to respond user's needs compared to the centralized models where all intensive tasks are transmitted to the cloud servers. Similarly, the overall performance of each application is enhanced, ensuring the achievement of the required latency and high accuracy. Moreover, security is guaranteed avoiding excessive network traffic, these transfers frequently being victims of intrusion attacks.

Last but not least, in Smart Cities the continuous measurements made by sensors stored for long periods of time, offer valuable information to different entities. These could obtain diverse type of benefits depending on the area they work. Citizens' pedestrian activity patterns can give an insight of typology or essence of each street, building, urban area, etc. This information used in an intelligent way can enhance these entities' performance. For instance, strategical locations could be discovered following this methodology, having a positive impact on these entities. Long-time sensor readings can give an insight of future natural and social incidents. Attending these with enough precocity would avoid major disasters. Similarly, conflicting areas could be uncovered and entities like local police must focus their effort on these areas when it comes to security. Different urban conflicts should be resolved with less effort anticipating them, like strikes, football hooligan battles or other criminal acts. In addition, citizens' life quality is improved by offering new opportunities with the information and knowledge provided by the tools presented in this thesis[6].

1.2 Theoretical framework

Smart Cities are entities that carry out intelligent management of resources and infrastructure to optimize their use, promote sustainable development and improve the lives of citizens. Predictive models make it possible to anticipate where and when a situation of saturation of public services (demand peaks) or underuse of them will occur. Temporal and geographical predictions on the behavior of users of public services are the key to effective management of energy distribution services (electricity, gas, district heating...), water management, garbage collection and even aspects related to emissions into the atmosphere and pollution, among others, and to anticipate demand. The identification of the different patterns of use of these services based on historical data facilitates their prediction and subsequent management. Furthermore, in addition to geographical and temporal relationships between data, there are also inter-relationships between the usage patterns of different services that graph-based data structures are capable of representing and quantifying. Similarly, pedestrians patterns of mobility could give an useful information about the location and timing of potential crowds.

Management and decision-making tools based on AI algorithms have great potential to offer new and more efficient services to improve people's living conditions. These tools are possible thanks to the collection of information from the physical environment in RT through the IoT and the fusion of a big quantity of data from heterogeneous sources (Big Data). Among the different data sources, in addition to the installed sensors, there is free access information or Open Data and the contextual information associated with the data, mainly space-time but also the relationship between the different variables. The collaborative use of all of these type of data would bring various benefits to the citizens, starting from reduction of energy and pollution to optimized public services. Security would also be enhanced with an adequate extraction of knowledge from the sources already mentioned.

The use of sensors, actuators or other type of IoT devices offer useful information to the users, RT response and provide information to higher level computing centers for more complex tasks. On the other hand, DL models require high computability for the training phase and a big amount of available memory to store their parameters for the latter inference. Consequently, IoT devices cannot store such amount of data and train deep models, needing to adjust data and model sizes or send these assignments to more powerful devices. These could alleviate the issue of lack of resources faced by IoT devices, offering them an alternative to carry out a diversity of tasks[1].

In an architecture of different computing centers in which we can distinguish different layers depending on the computational complexity of each component of the network. There are many criteria for dividing layers and we can find different sparsification levels in these layer classifications. All of these, include

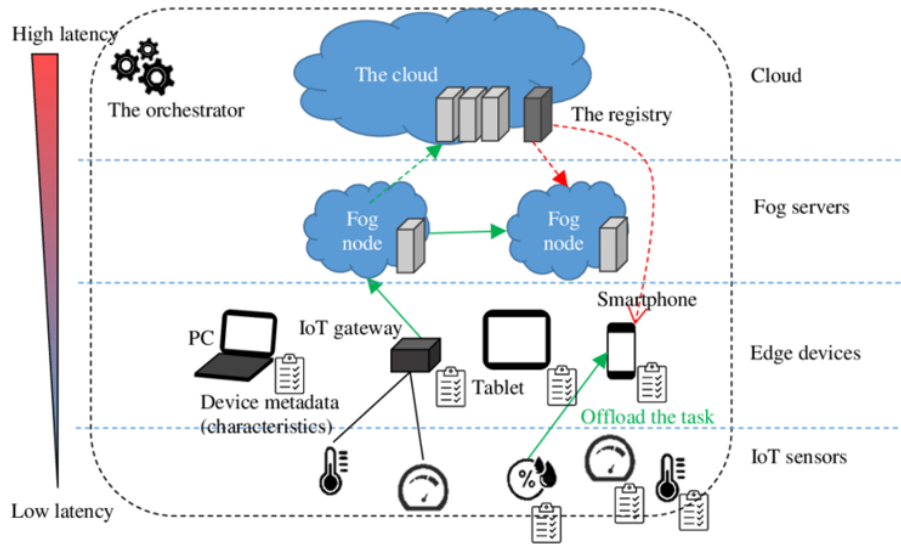


Figure 4: Different layers of a network

Source: PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-task-offloading-flow-and-the-role-of-the-orchestrator-It-consists-of-the-following_fig4_347134683

a cloud level in which there is a general resource-rich network remote servers with ability to store, manage and process data. It offers appropriate automation tools that could be useful in many cases. Moreover, it is equipped with computing resources that are totally configurable, with possibility of release with a minimal management effort.

In contrast, the lowest level where sensors and IoT devices could be found is referred as IoT layer. These devices are equipped with restricted computing capabilities but offer immediate response to the users. Meanwhile, between them we can define other layers such as Fog and Edge layers with computers provided with higher capabilities than the previous ones, but with less processing capacity and memory than the cloud. However, they are usually located close to the IoT devices, which is a clear advantage for various reasons such as latency and security among others.

The IoT devices are not equipped with as much complex architectures as other higher-level computing centers such as cloud servers or Edge Datacenters. Although the level of compression is very high on these devices, they still lack of sufficient space to store large memory and powerful [7]. As a consequence, they are not able to carry out tasks that are far more complex than giving a

simple measurement or giving an immediate response to a certain input. These tasks include calculating derivatives of first and second order or iterative loops that can carry a very long time to execute in a simple processor. Oppositely in a high capacity multiprocessor computer, parallel computation and greater speed ease the execution in a considerably faster way than in the previous case. The storage of millions of parameters needed to describe DL models is viable in these devices equipped with sufficient memory. On the contrary, due to the high number of components needed to constitute DL models is not possible to carry out inference process of such models in IoT devices, not at least in their original version.

Instead, at the other end there is usually a cloud server without problems of memory availability and very high processability in every networking environment. The weakness of using this alternative is that the transmission of information from IoT devices to the cloud carries certain time delay and possible loss of information through the network[8]. This loss could be a consequence of loss of connection or other type misinformation of messages. Not only information could be incomplete or with errors, but also it would be a potential victim of intrusion attacks. According to [9] in the digital world, malicious activities that violate the confidentiality, integrity, or availability of data and devices are known as intrusions. The vulnerability is higher when the time that the information is exposed through the network increase, and that is the reason why transferring to the cloud confidential information is not the appropriate election. On the contrary, IoT devices do not expose information through the network when they attend any task, becoming the safest option. The latency required by many applications also preclude the use of cloud as the final computing center due to longer time delay. The essential requirement of RT computation is the immediate response, and that is impossible to achieve using cloud computation. Consequently, there is no other choice than completing these tasks in IoT devices or at the most in the closest Edge Datacenter.

However, the ML architectures designed for carrying out diverse type of tasks could be modified while preserving their efficiency and accuracy to a large extent. Reduction of the amount of parameters that represent the models is a possible way for achieving it, reducing the information used for representing these parameters or by modifying the structures of these architectures [4]. By this way, a reduction in size of these models is achieved, reducing the size of data needed to represent their components as well. Finally, the modified versions of models could fit in the mentioned resource constrained devices and they could avoid the lack of memory problem. Although the processors of these devices will continue being much weaker than those in the cloud and Edge Datacenters, this modifications facilitate the execution of diverse tasks without offloading to more powerful devices with the advantage that this implies in terms of immediateness of response and security.

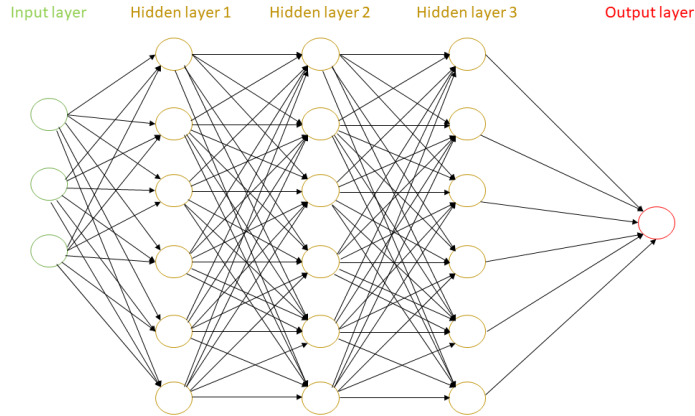


Figure 5: Graphical representation of a Neural Network

Among supervised ML architectures, one of the most used and successful is the NN. NNs are composed of neurons and connections. Each of them have assigned a weight that is modified as learning goes forward. If the weight increases or decreases, the signal strength of a given connection does the same. A threshold should be established for each connection such that a signal is sent only if the aggregated signal surpasses that threshold. NNs are trained using labelled instances, each containing a known input” and output(label), forming probability-weighted associations between this pairs, that are stored within the data structure of the network itself. The training of the network from a set of examples is carried out by calculating the difference between the inferred output of the network (prediction) and a target output, this difference being the error. Consequently, the network adjusts its weights using the error value and in accordance with a learning rule. After several iterations, on which many refinements of the weights take place, the NN will produce the desired output that should be similar to the target output. The training would be terminated based on certain criteria. A graphical representation of this architecture is given in Figure 5.

One of the most used and effective ways to reduce the dimension of these networks is the use of techniques such as pruning and quantization. The first one consists of removing elements (neurons or weights) that have negligible contribution in the final result or prediction. The criteria followed to decide which parameters have to be removed is often complex. On the other hand, quantization involves replacing different data types to reduced width data types, by transforming data to fit into new data types’ shapes. As a result, resultant networks are able to emulate the performance of the original ones and even improve these in some cases in which overfitting issues were hindering their pre-

dictability. Additionally, by reducing the width of the data, IoT devices could face the storage issue mentioned above and collect datasets and parameters of these networks in constrained memory sizes.

Having knowledge about which of the elements of the original network are the most adequate to prune is not straightforward. A good criteria to select components to prune from the original network is the use of the derivatives of the Hessian matrix [10][11] or Taylor expansions [12][13] of first order to approximate the change of loss in the objective function as an effect of pruning. Attending the magnitude of the components or weights of each layer of the network [14][15] is a good criteria to lighten the original layers of the network as well. Low-rank decomposition for convolution layers as well as fully connected layers were applied in several works [16][17][18].

Other alternative to carry out ML tasks with latency requirements successfully and avoiding excessively long network transmissions that take place when the tasks are sent to the cloud, is the offloading of tasks to near-by available computing centers equipped with sufficiently strong processors and large memories. When IoT devices are assigned to execute certain computation but they are not equipped with sufficient resources or they are overloaded, they have the opportunity to transmit through the network their assigned task to other more powerful devices thanks to the interconnectivity between different computers and gadgets. The destinies of these offloaded tasks could be other Edge Datacenters that offer the possibility to temporarily store data. Less interesting is the option of transferring the tasks to the cloud server, due to the reasons mentioned above. In some cases, there are other gadgets in the IoT layer that have sufficient capabilities to respond certain type of tasks' needs, being potential destinies for these type of jobs.

Proper offloading strategy is crucial to avoid situations in which in an architecture certain computers absorb all the tasks of the end-user devices surrounding it. As a result, several tasks may not be executed or may be executed with certain delay. To avoid such a situation, load balance between computing Datacenters should be guaranteed and all tasks should be successfully executed. In some scenarios in which latency requirements are much more flexible, cloud servers could provide the same functionality of Edge Datacenters, with higher computability but with longer network transmission. Although this would carry a relaxation of load between Edge Datacenters, these strategies are more prone to suffer from intrusion attacks and network failures. In contrast, Edge Datacenters have sufficient available memory and processability for the majority of the assigned tasks and they are located closer to the users. As a consequence, shorter time delay and higher security and safety have a positive impact on the final applications' performances. With an optimized orchestration strategy, tasks should be equally distributed between all available Edge Datacenters, assigning more tasks to the ones that have more free memory space and whose processor is less loaded.

According to the traditional time series analysis, various analysis can be performed on the data such as plotting correlation and autocorrelation plots, time domain vs frequency analysis, state estimation over time, etc. [19]. classify into multilinear regression models (ARIMA), models based on NNs (CNN, RNN) [20][21] and GNNs [22]. GNN applications applied to spatio-temporal predictive models are listed below:

- Since these networks are capable of capturing spatial and temporal patterns of the graph, they can be used in various applications in intelligent transportation systems (ITS), such as route planning, navigation, and traffic control and management.
- Due to the ability to handle large amounts of information, spatiotemporal GNNs can be used for forecasting, conflict prediction, and pandemic prediction.
- In RL, we can use these NNs to predict the future states of agents considering the social influence of other agents.
- These networks can also be used in successful inventory planning and logistics cost optimization for online marketplaces.

RL itself also offers potential to achieve knowledge from the environment, and apply this knowledge to make decisions[21]. RL is a general framework where agents learn to perform actions in an environment so as to maximize a reward [23].

The agent and environment continuously interact with each other. At each time step, the agent takes the optimal action (a_t) that maximizes the sum of rewards on the environment based on its policy $\pi(a_t | s_t)$, where s_t is the current observation from the environment, while receives a reward r_{t+1} and the next observation s_{t+1} from the environment depending on the action taken(a_t). The goal is to improve the policy so as to maximize the sum of rewards (return). The dynamism of many real world environments make this DL branch very suitable to use in such environments. The constant changes that may appear in a Smart City or a Smart Building have a consequent downgrade of performance in the models that work on them if these constant variances are not considered. These constant updates that will be the nature of learning process of the RL techniques, ensure consideration of the latest modifications of the environment in which the application takes place. As a result, a better representation of the problem is guaranteed and a better policy should be obtained following this learning process.

When the environment that is to be learned is simple, an optimal policy that guarantees good results is easily achievable. However, when there are many possible states and several type of actions, there are more complex approaches

that are better suited to the problem at hand. Combination of NNs and Q-Learning, which is a model-free RL algorithm that determines the value of a given action in a given state of the environment, was proposed as an alternative to the original version, offering a new way of obtaining the optimal policy. The Deep Q-Network (DQN)[24] is composed by two NNs, the first one represented by the θ parameters, for estimating the Q values of the action a and the second one represented by θ' for estimating the Q_{t+1} values of the next state s_{t+1} and action a_{t+1} . The learning process takes place in the first network or the principal network, and after a number of iterations the values obtained are copied to the second network which has been fixed for those iterations. Using the Bellman optimality equation (see Equation 1) as a repetitive refresh, convergence of the Q function is ensured.

$$Q_{i+1}(s_t, a_t) = E[r + \gamma * \max_{a_{t+1}} Q_i(s_{t+1}, a_{t+1})] \quad (1)$$

Using the NNs approximation can be done using θ parameters and minimizing the loss function. Here, the The discount factor γ states the influence of future rewards.

$$L_i(\theta) = E_{s_t, a_t, r, s_{t+1}} \rho [(y_i - Q(s_t, a_t; \theta_i))^2] \quad (2)$$

$$y_i = r + \gamma * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) \quad (3)$$

From a broader perspective there are many possible applications that use of the information derived from IoT devices. In a Smart City, sensors act as data accumulators, sensing different events of the environment in which they are located, such as traffic incidents or pedestrians' mobility, and transmitting these data to more powerful servers. The measurements of these devices should be stored in a Datacenter or a remote server from which valuable information would be obtained to enhance citizens lives. Nonetheless, the continuous activity of these devices needs a constant supervision and an appropriate energy management. The simplicity of these devices carries frequent breakdowns and other problems that needed to be supervised and repaired by experts. As if that were not enough, vandalism attacks occur often in conflicting areas, becoming essential a constant supervision of these gadgets.

Predictions about citizens' lifestyles and mobility patterns could be useful for different end users. The sensor measurements data stored for a certain period of time could give an insight about the pedestrians trends through different metropolitan areas. This tendencies might have variations during the day and throughout the week. After collecting data throughout a certain period of time and transmitting it to a remote server in which data it can managed and stored, data processing is essential. This involves elimination or replacement of incomplete and erroneous data, translation of formatted data and detection and elimination of outliers. As a consequence of a correct understanding of data,

different applications could be pioneered.

Different classification of sensors or group of sensors could be done by applying supervised ML algorithms or unsupervised ML algorithms. In the same way, bigger urban areas could be classified such as streets or buildings. These could benefit citizens lives by offering the possibility of knowing with a series of simple measurements the typology of each of the mentioned entities, giving an insight of what they could find when they access there. Moreover, by using ML algorithms detection of anomalies in large data is feasible[25]. In a similar way, different entities could obtain benefits from these classifications. This is because knowing in RT the patterns of use of streets or buildings would ease the detection of conflicting areas, for instance. The council could detect touristic zones, so that in the future these attractions should be emphasized for new visitors.

These classifications could be done for processing data for a short period of time or a longer period. In the first case, changes in the mobility of citizens could be detected in RT, and consequently optimal decisions should be made to guarantee security and good standards of living of citizens. In the latter case, some entities such as merchants would obtain the population mobility patterns of different urban areas and decide strategically where to locate their shops. Moreover, the activity of these customers throughout the day and for different days of the week could give an useful information for establishing the optimal stores' opening hours.

1.3 Objectives & Research Methodology

In this section general and specific objectives are described and the works that were focused on these are referenced. In addition, the research methodology that have been followed in these works is described.

1.3.1 General objectives

These are the main objectives that are addressed throughout the thesis, framed in the ML Area following two main lines:

- **Algorithmics/Computation.** Algorithms belonging to different categories of ML, including supervised learning, unsupervised learning, and RL, have been studied in depth. New variants of algorithms have been developed that introduce new knowledge. We tested our alternatives with methods from the literature on known datasets from [23] and using a well-known simulator [26] in the field. In this way, the proposed algorithms were competitive with regard to the accuracy and other performance metrics against the best literature methodologies proposed so far.
- **Applications.** New applications have been proposed from data captured by small sensors stored for long periods of time. Through the work developed in this thesis, valuable new knowledge was obtained and possible applications along these lines were identified. To do this, we worked with open source data from the city of Melbourne [27] & Madrid [28] in the context of the Building Prediction work.

1.3.2 Research methodology

The research procedure in all the works that form this thesis has been the following:

1. Notice about a deficit or problem (a suboptimal or improvable alternative) in the literature.
2. Propose an alternative or solution to the deficiency in question in some aspect of ML.
3. Test new variants/alternatives from an empirical perspective, comparing them with competitive methods from the literature.

The programming of the computational/algorithmic part of all the works has been using Python programming language [29]. To be specific, Python has been used to implement the algorithms used to outperform the results of NN reduction techniques [4], for the algorithms for the appropriate task offloading decision problem, and for the algorithmic part of the application building prediction. The packages that have been useful in computing processes are the following: Pandas [30], Numpy [31], Tensorflow [23], Keras [32] and Scikit - Learn [33]. For the task offloading paradigm, the experimentation process was

developed in the PureEdgeSim Edge Computing simulator[26].

The hardware environment in which all of our development work was carried out is a $\times 64$ Ubuntu 20.04.4 LTS Operating System equipped with an Intel Core i7-11850H working at $2.5\text{ GHz}\times 16$ and 32 GB DDR-4 RAM and a NVIDIA T1200 Laptop GPU (driver version: 510.47.03, CUDA version:11.6).

In the experimental phase of our work, databases from the Tensorflow [23] and UCI [34] repositories have been used. Open data from the city of Melbourne [27] & Madrid [28] has also been used.

1.3.3 Specific objectives

Among all the aforementioned methods and strategies that would be potential techniques for obtaining novel insights into citizen patterns, the objectives specified in this Section summarize the direction of the thesis. The general purpose is to complete computationally expensive tasks, such as inference and training DL and ML models on resource-constrained devices, without the need to send them to remote servers. Furthermore, these small devices should be a valuable source of knowledge, as they continuously measure real-world events. As a consequence, data stored over a longer period of time must be used in cooperation with readings from other devices and other information to gain valuable insights into Smart Cities and Smart Buildings.

- The first specific objective of the thesis was the optimization of algorithms to adapt to the characteristics of devices with limited resources, covering the general algorithmic/computing objective. This optimization of the ML model for processing on a device with limited resources (hardware) will be carried out by shrinking the model in memory, adapting the inference time to the response time needs of the application and reducing energy consumption of the IoT device while maintaining the accuracy of the results as much as possible. This objective was met in [4].
- The second specific objective of the thesis is the processing of information, making decisions and acting on the active systems of the technical infrastructures of the Smart City through “Intelligent Sensors/Actuators” (Edge Intelligence and On-device Intelligence) using techniques of Edge Computing (EC) related with the algorithmic/computing general objective and met in [5]. The training of the ML models to distribute the workload in the Edge will be carried out in the cloud (Cloud Intelligence) which will subsequently be optimized for computing in embedded systems (Edge Computing).

- The third specific objective of the thesis is the improvement of predictive ML models applied to spatio-temporal data represented in the form of relational graphs and their validation in the field of optimization of public services offered in the Smart City (energy consumption, water, garbage, pollution, etc.) connected with the second general objective. The research to be developed will focus on the adaptation and improvement of these models to predict the use patterns of public services in the city and buildings with the aim of optimizing their management by efficiently incorporating the spatio-temporal interrelationships existing in the data. This objective was met in [6].

Reduction of Neural Networks

As briefly mentioned in the previous section, one of the drawbacks of the need to serve different tasks in RT at the IoT layer is the lack of computational resources. Sensors, actuators and other types of IoT devices can act immediately to provide a response in this type of applications but have several limitations in terms of computability and memory. As a consequence, it is not feasible to run some of the ML models that are designed to address the needs of RT applications on these weak devices. Especially, DL networks have billions of parameters needed to represent their structure. These devices are not prepared to perform either their training or subsequent inference, which needs to store billions of parameters that represent these deep models.

For this reason, according to the first general objective, a refinement of the reduction techniques included in ML has been carried out, realizing the possibility of completing ML complex algorithms in resource-constrained devices. To do this, the data must be preprocessed, managed, and compressed to ensure that the data size is compact enough to fit in the devices' memory.

The dimension of different ML algorithms and especially DL algorithms is such that in most cases it is not feasible to run them on IoT devices. The number of parameters to represent these architectures is too high and the values of these parameters need good precision to obtain desirable model performance. At the same time, the devices in question with limited resources cannot store all the data with high precision, so some data need to be ignored or a lower precision needs to be set to represent the parameters.

To achieve the goal of adapting DL models to IoT devices, pruning techniques are essential to remove some of the elements from the original networks that have a minor impact on the output. These elements are redundant due to the negligible effect they have on the predictability of the model. Different elements of the original network can be removed, whether they are connections between different neurons or the neurons themselves. The decision of which items should be removed is not straightforward and a good policy is needed to make this decision. According to the first specific objective, our intention was

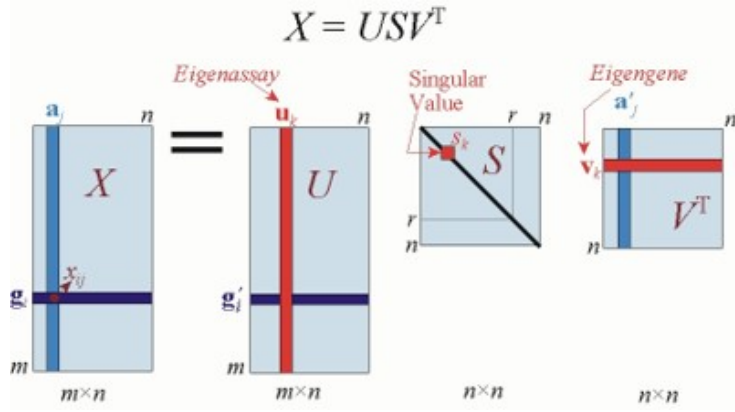


Figure 6: Single Value Decomposition
 Available from: <https://webdocs.cs.ualberta.ca/~rgreiner/R/OLD-BiCluster/SVDApproaches.html>

to outperform the most promising SoA methods by offering new alternatives that include the knowledge demonstrated in relevant works.

Deficiency or problem Single Value Decomposition (SVD) was a valuable methodology to perform this reduction by decreasing the number of parameters of the original networks. In an FC layer that has m inputs and n outputs neurons, activation $a \in \mathbb{R}^n$ of the layer with n nodes is represented as

$$a = \mathbf{g}(\mathbf{W}^T \mathbf{X} + \mathbf{b}) \quad (4)$$

where \mathbf{X} describes the input to the layer, and $\mathbf{g}()$ describes any feasible activation function. FC layers connections are represented by a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ and a bias vector $\mathbf{b} \in \mathbb{R}^n$ where each parameter in the weight matrix \mathbf{W} is $w^{ij} \in \mathbb{R}$ ($1 \leq i \leq m, 1 \leq j \leq n$), and bias matrix \mathbf{b} is $b^j \in \mathbb{R}$ ($1 \leq j \leq n$). Using low-ranks factorization original matrices are decomposed. The decomposition is applied to the weight matrix \mathbf{W} after the entire model has been trained. The SVD approach decomposes the weight matrix \mathbf{W} as $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{V}^T \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\mathbf{S} \in \mathbb{R}^{m \times n}$ is a diagonal matrix.

\mathbf{S} is a singular diagonal matrix where the most significant singular values go from the top left to the bottom right in descending order. The truncation process consists of maintaining k most significant values in the decomposed matrices. By this way, the first k rows of \mathbf{U} and the first k columns of \mathbf{V}^T are kept. In order to achieve a higher compression rate Sparse Low-Rank Method (SLR) [18] is a valid alternative. Instead of using rank k reduced rank rk is used. By this way, first rk rows of \mathbf{U} and the first rk columns of \mathbf{V}^T are kept, being rk lower than k . Conserving only the most significant rows (rm) and columns

(rn) from each column and row from \mathbf{U} and \mathbf{V}^T , respectively, following a cost criteria, even a higher compression is achieved. This criteria is defined in the next paragraphs.

In a Convolutional Neural Network (CNN) there are two main type of layers. Some are Fully Connected (FC) Layers and some other Convolutional Layers. FC layers are those in which each node of each layer is connected to each node of the next layer. All nodes in subsequent layers are interconnected in these types of layers. On the contrary, in convolutional layers the operation of convolution takes place, which is a sliding scalar product, where the kernel moves along the input matrix and the scalar product is obtained in the same way that occurs with vectors. A higher flexibility of convolutional layers in learning is due to the fact that not all input nodes of a neuron are connected to output nodes. The number of weights per layer is also much smaller, which is specially useful with high-dimensional inputs, such as image data[35].

Both fully connected layers and convolutional layers can be pruned, but it is much more effective in terms of accuracy, time and energy efficiency to prune the former as shown in [36], which contributes to lower losses in the prediction ability with the same parameter reduction rate. Convolutional layers are typically placed in the first positions in DNNs and are more sensitive than those placed in the last positions in many cases. With the dispersion of SVD matrices a low compression rate can be achieved without large losses in precision. The criteria for sparsification can be diverse, being the compression rate defined in [18] the optimal one in theory, the relationship between the parameters necessary to define the sparse decomposed matrices and the parameters of the matrix of the original weights. In our work we aim to overcome the precision of the resulting network originated from sparse matrices, and the proposal is the consideration of relevance in the final decision of more than one layer.

The importance of a neuron is defined by whether or not there is a change in network performance after removing it. Let c be the default cost of the NN with the original trained weight W estimated for the p training samples, computed using any loss function. Let \hat{c} be the cost value of the network with sparse weights $\hat{\mathbf{W}}$. By truncating with reduced range rr a specific row of $\hat{\mathbf{U}}$ or a column of $\hat{\mathbf{V}}^T$ we have the absolute change in cost is or os . These are calculated as follows:

$$is_i = |c - \hat{c}_i| \tag{5}$$

$$os_j = |c - \hat{c}_j| \tag{6}$$

In this way, the rows and columns with the highest cost are considered the most important rows (rm) and columns (rn) of each column and row of \mathbf{U} and \mathbf{V}^T , respectively.

Although the use of the SLR method is a valuable tool for pruning a weight matrix, it treats the layers of an NN independently, without considering any

correlation between them. Each layer of a NN is interconnected with each other, and there is no doubt that some of them are more relevant than others, or have more impact on the output of the network. But these layers, since they are not isolated, should backpropagate their relevance to the previous layers as proposed in [37]. As a result, the relevance of each neuron in the final decision is the composition of weights that are interconnected to the relevance of the corresponding element of the Final Response Layer (FRL).

Proposed alternative To overcome the methods available in the literature, we chose to mix the principles of the SLR method [18] and backpropagation[37]. The sum of the relevances corresponding to each layer is given by the Equation 7.

$$s_k = |\mathbf{w}^{(k+1)}|^\top |\mathbf{w}^{(k+2)}|^\top \dots |\mathbf{w}^{(n)}|^\top s_n \quad (7)$$

The relevance of each neuron in the final decision is the composition of weights that are interconnected to the relevance of the corresponding FRL element. The absolute values of the weights that are connected to each of the FRL neurons are multiplied by their relevance in the FRL.

$$s_{k,j} = \sum_i |w_{i,j}^{(k+1)}| s_{k+1,i} \quad (8)$$

The equation 8 shows the relevance of the j th neuron in the k -th layer, which propagates the relevances of the neurons in the subsequent $k + 1$ -th layer that are connected to it .

By introducing this idea into the SVD matrices, keeping only the most relevant rows of the U matrices, we can consider only the most relevant neurons of each layer. The procedure in the non-FRL FC layers is similar to the original SLR method except for the scattering of the U matrices where the relevance propagated through subsequent layers is considered to determine the most relevant neurons.

As a consequence, a better consideration of the relevance of each neuron in each layer is achieved, due to the correlation between layers of a network. In this way, pruning the irrelevant components of each layer can be optimally achieved by retaining only the most significant parts layer by layer.

Testing the proposed alternative Our proposed variant, SLRProp [4], was tested against the original SLR method. Both have been tested on well-known open source image recognition datasets Cifar10, Cifar100 and MNIST obtained from Tensorflow [23]. All of them have been trained using 10,000 default test images and 50,000 and 60,000 training images for the Cifar and MNIST datasets, respectively. To show their effectiveness on sensor-related datasets, they were also applied to the room occupancy estimation dataset [34]. In this case, 1000

samples were used for testing and the rest (9129 samples) were used to train the network.

In each case, we opted for establishing the same reduction rate (0.5) and sparsity rate (0.5) defined in [18], and we tested each variant with different rank k , which determines the number of columns and rows kept in the sparsified matrices. We incremented the rank k until the performance metrics were equal to the ones obtained by the original network structure. In the testing phase 10 different seeds were established for testing each methodology in each dataset. The network models pruned were VGG16 applied on Cifar10 and Cifar100 datasets, Lenet5 applied on MNIST and finally a model consisting of 3 FC layers for Room Occupancy Estimation dataset.

Evaluation metrics used for determining which of the methods used was best for keeping the performance of the former network as high as possible were the accuracy vs. compression rate, AUC vs. compression, recall vs. compression, precision vs. compression, and specificity vs. compression, where the compression rate was defined in [4]. This last metric determines the ratio of the number of parameters between the sparse decomposed matrices and the weight matrices of the original network. Accuracy determines the goodness of the classifier. Recall provides the probability that an instance classified as positive is actually positive. Precision states the probability that an instance belonging to the positive class is classified as positive. AUC is the area under the ROC curve, that is, a graph that shows the performance of a classification model at all classification thresholds. What is plotted on the curve is the FPR and TPR on the x and y axes, respectively, whose definitions are given in Equation 12 and 13. The definitions of the rest of the metrics mentioned above are given in Equations 9–11, where TP, TN, FP and FN represent True Positives, True Negatives and False Positives and False Negatives, respectively. We use the compression rate of the previous FC layer of FRL to verify the accuracy of the resulting network under different compression rate regimes.

$$Accuracy(Acc) = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

$$Recall(Re) = \frac{TP}{TP + FN} \quad (10)$$

$$Precision(Pr) = \frac{TP}{TP + FP} \quad (11)$$

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN} \quad (12)$$

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \quad (13)$$

$$Specificity(Spec) = \frac{FP}{FP + TN} \quad (14)$$

When the problem is multi-class, the evaluation of these parameters is carried out in the following way. For each class, all instances that do not belong to the class in question are considered negative, and instances that belong to the class are considered positive. For each class there are Accuracy, Recall and Precision metrics, and the overall values of these metrics are obtained by micro-averaging (giving equal weight to each instance), macro-averaging (giving equal weight to each class), or giving different weights to each class, becoming an interesting alternative when the data is unbalanced. In our case, as the data was equally balanced we opted for using macro-averaging metrics.

Offloading of tasks in the Edge

The lack of resources continued to be a limitation on the solution given for the first specific objective. The algorithms and structures used to address user needs and application demands were modified and adjusted to the properties of resource-constrained devices. As a consequence, their performance suffered a degradation, less noticeable due to optimized reduction techniques but still significant in certain environments and applications.

On occasions when Edge Datacenters are located near these IoT devices, provided with greater capacities and greater memory than the IoT devices, these can be potential destinations for computing tasks that were not feasible to compute on the source devices due to memory and processability restrictions. According to the second specific objective, tasks have been offloaded to resource-rich platforms, processing information and making decisions at the Edge. In most cases, this is a better solution than compressing the original network architecture to achieve the desired precision for each particular application. Otherwise, if the accuracy requirements are not strict but an immediate response from the device is required, compressing the original structure would be a more suitable alternative, despite the short time required to send the information to the Edge Datacenters thanks to their proximity.

When multiple IoT devices offload their assigned task to the nearest Edge Datacenter, the latter may face the problem of overload. In this situation, many tasks may not be able to complete within the required latency, or simply will not be completed due to lack of resources. Alternatively, if there are other Edge Datacenters available within a reasonable distance, tasks that were not possible to complete on the source device could be offloaded to these alternative data centers. In other cases, some of the IoT devices themselves have small processors that can handle the simplest tasks. Consequently, for simpler tasks these latter devices could be good destinations to complete them.

Deficiency or problem When in a network environment there are different types of devices classifiable in different layers, such as the Edge layer, the IoT layer and the cloud, different types of potential download destinations will be available. For this reason, it is especially important to establish a good

download policy to make the most of your computing capabilities. Due to the processability and large memories offered by most Edge Datacenters, these must be preserved to perform the most complex and computationally heavy tasks. In contrast, simpler tasks and those that are supposed to react in an extremely short time should be computed on the source device or the closest device that can offer enough resources to complete them successfully.

The environmental characteristics that affect the offloading decision are multiple. Some of them are related to the characteristics of the devices and others to the nature of the tasks. The most representative properties that indicate the suitability of a device to solve a certain task are the available RAM memory of the computing device, the millions of instructions per second (MIP), the central processing unit (CPU) and the memory. On the other hand, the characteristics of the tasks that can affect the download decision are the desired latency and the file size in bits associated with the task.

Among the wide variety of tools to establish the optimal download policy, few investigations have chosen to use RL or GNN methods. The continuous changes in characteristics of many networking environments lead to the use of algorithms that take these updates into account in the learning process. The availability of Edge Datacenters can vary over time, while network traffic also has many variations RL is a useful branch of deep learning in which the agent continuously learns the properties of the environment online while receiving the reward obtained by performing a certain action. Consequently, the agent learns the changing environment where applications and tasks are developed and device and network characteristics updates are continually learned. For these reasons, the second objective of this thesis is to achieve a policy that optimizes the use of resources while ensuring the completion of tasks within the required latency. This goal can be achieved using DQNs, which are a good solution for diverse environments where there are constant updates and there are multiple possible states and actions.

Proposed alternative As an alternative to address the problem of the inability to attend to tasks at the IoT layer, an appropriate offloading algorithm was designed that improves the overall efficiency and performance of the systems and guarantees load balancing between the different computing centers.

To achieve these objectives, two different methods were applied compared to the known predetermined methodologies. In fact, GNN was applied as an architecture to emulate the network environment, where promising results were obtained. Furthermore, due to their suitability for constantly acquiring new information about the environment, DQNs were as promising as the previous ones in solving the problem at hand. Updated knowledge could be obtained by following this latest methodology.

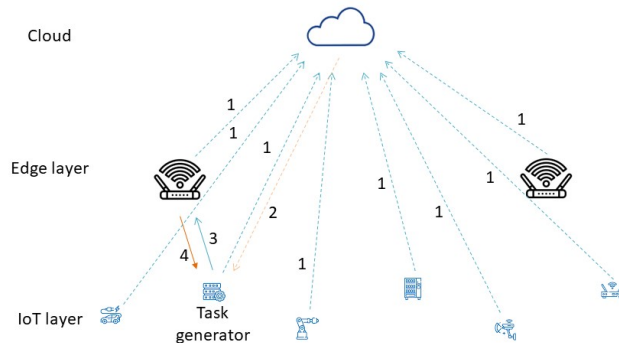


Figure 7: Overview of the training procedure of DQN.

In a DQN the environment is represented by all the possible states that would be in each case the real situation of the environment. In our case, the status of each device that makes up the network environment, including its actual availability parameters, task properties, and actual network traffic, would represent the state of the environment. The possible actions would be the possible decisions to download a certain task from its source device. Finally, the reward would be favorable or not if the task has been successfully completed satisfying the previously imposed requirements.

In the case of DQN, each device would send information about its status to the cloud (Step 1). There, having the state of the environment (s_t) based on the download policy $\pi(a_t|s_t)$, the optimal action a_t must be taken. If the task was completed meeting the requirements the reward would be 1, and 0 otherwise. In this way, an optimal download policy would be obtained after converging the Q function. Finally, the obtained Q function would determine the optimal download destination in the cloud after each device sends its state to the cloud, and this returns the message to the task generator indicating where to download your assigned task (Step 2). After sending the task to the target device (Step 3) and completing the task on this device, the results will be sent back to the source device (Step 4). A graphical description of the DQN training process is provided in Figure 7.

In addition to RL alternatives, another Deep Learning architecture that adapts well to the network environment where different devices are interconnected with each other is a Graph. A Graph is composed of a set of nodes(V)

and edges(E), represented by $G(E, V)$, whose data structure is non-linear. The edges connect any two nodes in the graph. Any system consisting of space and time structural relationship information can be contemplated as a space-time graph. NNs can deal with static structures and time-varying features of the graph.

A graph is represented by the pair $G = (V, A)$. The V represents the set of nodes and the A the set of edges. The nodes represent the elements of the system and the edges the interconnections between them. A valued graph is referred to a graph G that has assigned a numerical weight W_{ij} to each edge (i, j) . There is a possibility of weighting the nodes as well assigning a value W_i to each node i .

A GNN is a sort of NN that works directly with the graph structure. Characteristics of interconnections between devices can be represented by edge characteristics, and node characteristics can be characteristics of the devices themselves. Otherwise, edge features may also represent another type of relationship between devices. Working with a GNN would allow us to know the properties of each situation in which tasks originally assigned to IoT devices are offloaded to other devices, and to inform the user about the optimal download destination on each occasion.

The features mentioned in the previous paragraphs can be reflected in the node features, because each node in the graph can represent each computing device on the network. Some of these computers may have assigned a task that must be completed within a desired time period, and this task would also be represented by the node itself. Each node is interconnected with each other in the network structure and the edges between the nodes can reflect each network connection, the characteristics of which must be representative of the application in question. Because of this, connection offloading rating could be a good predictor of how good a download path is at transferring a certain task at a given time on the network. This score can be defined as the number of tasks successfully executed using the offloading path, divided by the total number of tasks offloaded using the path, using a predefined offloading policy in a training process. Using this new parameter, all network connections can be classified based on their suitability for each situation, informing end users about them.

These features describe the network structure in a parallel graph network structure. The input of the network would be composed of the characteristics of the tasks and devices that make up the network structure, and the output, the optimal download destination for that task. Consequently, a GNN could have the ability to adapt its neurons and weights to achieve the desired output with each input to the network. The ground-truth data for deciding which is the optimal offloading destiny in each case has to be established as well. This could be obtained by following any of the previously defined and tested methodologies, that is, by following a certain policy, the download destination obtained would be the optimal one in each case. As a result, the ground-truth data would be used in the training phase and consequently the NN would adapt its components.

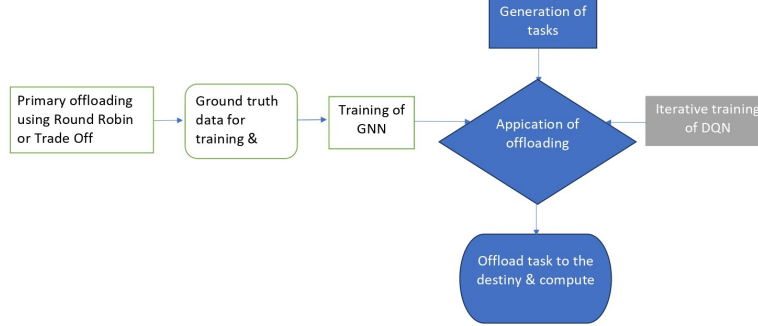


Figure 8: Overview of the entire process.

Once the entire training procedure of the default algorithm has been completed, the GNN would use as ground-truth the offloading decisions and the output generated in the previous step, and carry out the training process after defining the offloading rating for each network connection as edge feature. The training of the GNN would be carried out in the cloud. Finally, the download decision would be made by the cloud after each device sends the message with information about its status, and the cloud would return the message to the task-generating device informing about the download destination. After sending the task to the target device and completing it on this device, the results will be sent back to the source device. An overview of the entire process is given in figure 8.

Testing the proposed alternative We reproduced our proposed alternatives in the simulator [26] and we tested them against the default offloading policies of the simulator. We established different numbers of Edge Datacenters, IoT devices and the cloud, and we repeated the experimental process under different conditions. We opted for establishing 10, 20 and 30 end user devices, forming the IoT-Edge Layer, and we repeated the experiment 3 times. These devices were dynamic and their range of motion was limited to an area of 200x200 units. The Fog-Edge layer was composed of four data centers each of them located symmetrically in the covering area. Each of these Edge Datacenters was covering an area of 100x100 units. Finally, there was a resource rich cloud platform offering higher computability and memory.

Each of the IoT devices were interconnected with each other. In this way, interconnections between them were feasible. Similarly, each of these IoT devices was connected to the nearest Edge Datacenters and they were all connected to

the cloud. The orchestrator of the decision to download was the cloud. It was equipped with 200 cores, with 40,000 MIPS and 16 GB of RAM and 1 TB of memory. The Edge Datacenters were equipped with 10 cores, with 40,000 MIPS and 16 GB of RAM and 200 GB of memory. Its idle power consumption was 100 Wh with a maximum consumption of 250 Wh.

The operating system of the IoT devices was Linux and they had an architecture of $\times 86$. These devices had dynamic behavior in some cases, with a speed of 1.8 m/s. The type of network connection used to interconnect with the rest of the devices was WiFi with a bandwidth of 1300 Mbits/s, with a latency of 0.005 s. There were 5 different types of Edge devices with different battery, mobility pattern, memory and processors. Each of them generated 4 different types of tasks, with different generation rates, latency and file size length.

In the experimental process we considered the following parameters: energy consumption, tasks executed in each layer and success rate, that is, number of tasks successfully executed divided by the total number of tasks. Task failure could be due to different reasons, such as lack of available memory, violation of latency constraints, or network traffic congestion. In each case, the potential download destinations were all types of devices, IoT devices and Edge Datacenters, IoT devices and cloud, Edge Datacenters and cloud, IoT devices themselves, and Edge Datacenters themselves. 6 different tests for a fixed number of IoT devices.

Furthermore, we considered the download distribution between different types of download destinations when the number of IoT devices was 10, 20 and 30. Finally, to know more about the performance of these destinations, for a different number of IoT devices we obtained the metrics of performance for the devices that make up each layer.

Cooperation of Edge devices

The cooperative use of IoT and Edge devices can alleviate different problems in a complex system such as a smart building or a smart city. According to the second general objective, useful information could be obtained by acquiring, storing and analyzing the readings and measurements obtained through IoT devices, such as sensors or other types of gadgets. The positioning of these devices would be a key factor in deciding what information is intended to be obtained. Placing sensors in urban areas would be useful to learn more about crowd evolution in metropolises, or placement in indoor areas could be useful to address capacity issues or to understand mobility patterns, behaviors and attitudes of users.

Deficiency or problem The placement of sensors in different buildings would be a source of information about the mobility patterns of pedestrians and the typology of these establishments. Likewise, different identifications can be made

with these trends. There are big differences between the mobility patterns of hospitals or shopping centers frequented every day and schools or offices, which people rarely access on weekends. At the same time, there are fluctuations throughout the days, with peak hours in some cases. ML and DL tools would be useful to gain valuable insights from these readings obtained from IoT devices and stored over a long period of time, either in an Edge Datacenter or on a remote server. Following this proposal, the third objective of the thesis is to devise an innovative way to identify building types from sensor readings. There was no other work that offered this type of knowledge in the way we propose, thus filling the gap found in the literature.

Proposed alternative In a metropolitan area the number of devices located to perform uninterrupted measurements is very large. Consequently, information overload is a potential drawback encountered in these scenarios. Excess information would make it difficult to train ML algorithms, making it difficult to obtain novel insights into the properties of different urban areas. Due to this excess of features obtained by sensors and other IoT devices located in Smart Cities and Smart Buildings, a dimensionality reduction is often essential for an adequate training process of ML algorithms. Consequently, the reduced size of the datasets better adapts to most models, facilitating the training process, obtaining better results in the inference process and achieving more precise knowledge.

Readings from sensors located outside several buildings would give an idea of pedestrian mobility patterns between this type of infrastructure. Consequently, knowing the frequencies of access to these buildings it is possible to achieve a classification of the buildings based on their nature or typology.

Open access data is a valuable source of information from which new insights can be gained. There are different repositories that store measurements of various types of nature from different cities around the world. This data is open to all users and becomes a notable source. As if that were not enough, in most cases privacy problems make open access to large repositories that could be useful for different applications impossible. This is not the case with sensor readings, where the anonymity of citizens is fully preserved. In other fields, such as medicine, a large amount of data is not available due to privacy issues.

According to the third specific objective of this thesis, after reducing the number of characteristics of the data obtained by a large number of gadgets located in a Smart City to facilitate the subsequent training process, using an unsupervised ML algorithm and GNNs a classification of different type of buildings regarding their frequencies of access through different days and different time windows have been proposed.

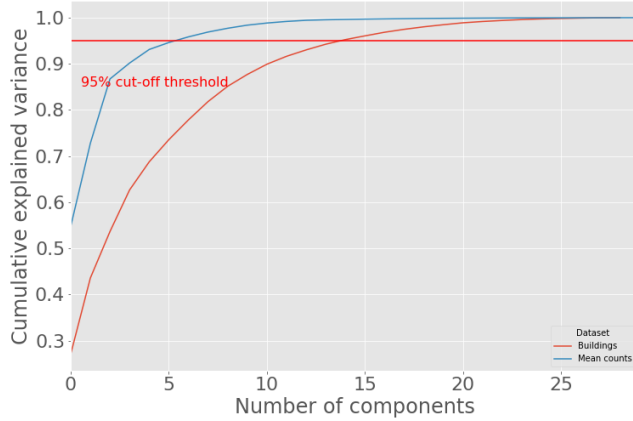


Figure 9: Cumulative explained variance for the Melbourne building feature based case and pedestrians with mean hourly mean count feature based case.

For these purposes, a feature size reduction is sought using Principal Component Analysis (PCA) as a tool, carrying out dimensional reduction of the original data set with a large number of connected variables while maintaining the greatest possible variance in the data set. This is a process of calculating the principal components of a set of points, that are succession of p unit vectors where the i -th vector is the direction of a line that best fits the data while being orthogonal to the first $i-1$ vectors, and using them to perform a change of basis on the data. We opted for regarding the proportion of accumulated explained variance and selecting the minimum number of components from which the increase is negligible. This proportion would be different for each applications, due to the differences on accuracy requirements. In this case we established a 95% variation that is explained by the number of components selected. Figure 9 shows that with 14 components the desired variation was achieved for the case based on building typology characteristics. Likewise, 5 components were sufficient to satisfy the requirement mentioned above for the case based on pedestrian activity characteristics. These different ways of dividing buildings are defined below.

Once the feature size reduction of the data is achieved, the next step would be the classification of different buildings based on their features. The nature of the problems leads us to use unsupervised ML methods, since there are no previously established labels for the readings of the sensors located in front of different buildings. Clustering methods are potential tools to divide these types of entities into different groups that discover natural patterns between instances. Group partitioning is carried out by achieving the greatest possible dissimilarity between elements belonging to different groups and the greatest

possible similarity between elements belonging to the same group. K-Means is an interesting and simplistic method to obtain good results in the clustering process with minimal effort. In this algorithm, each instance belongs to the group with the closest mean or centroid. At each iteration, these centroids are updated so that the geometric mean of the group is represented by the centroid.

In our case, the features used to represent instances (sensors located in front of buildings) were the type of buildings surrounding these sensors within a pre-defined radius on the first attempt, and their average activity over different intervals of time of different days of the week on the second attempt. Both groups would be a valuable source of new knowledge.

Similarly, GNNs and specially GCNs are a powerful tool to use in this scenario, given their similarity to the problem at hand. In this case, the nodes would represent the buildings or sensors located in these infrastructures. The edges or arcs that connect different nodes can have different types of characteristics that we would be interested in some of them. Distance would obviously be a good parameter, since most of the buildings of each type are located in the same cities, such as supermarkets, schools, sports stadiums, etc. Thus, the higher the distance between these nodes the lower would be the similarity between them. For this reason, the inverse of the distance between different nodes would be a suitable feature for edges.

In principle there was no labeled data, only sensor readings from different buildings. However, after performing an initial clustering of buildings and dividing them into different groups, a labeled ground truth data could be obtained and it would be feasible to use it in the subsequent GCN training phase. Using the inverse of the distance as edge weight, magnifying the similarity of the closest buildings, GCN could be trained.

In short, the GCN would classify each node of the network as part of each of the clusters made in the previous phase, each building corresponding to a typology. In the end, from the readings of the sensor located at the door of each building it is possible to arrive at the typology of these buildings.

Testing the proposed alternative We use data from Madrid and Melbourne to validate our proposed methodology. First, a PCA was performed to reduce the dimensionality of the problem and then clustering of different datasets was performed. To decide what would be the optimal number of clusters to perform the unsupervised ML algorithm, we chose to use the elbow method, which is a heuristic used to determine the optimal number of clusters. The method is as follows, plot the explained variation as a function of the number of clusters, and the point at which the graph flattens indicates the optimal number of clusters, enough to achieve the desired difference and consistency between the clusters. Finally, the clusters obtained were used as real data for

the GCN training process.

To validate these groups, different supervised ML methods were used. Using 10-fold cross-validation, inference accuracy was observed for each supervised ML model, applying 30 different seeds. This validation strategy divides the original set of training data into 10 smaller sets, each of them being used in one iteration for testing and the rest for training, completing 10 iterations. The selected supervised ML methods were Supported Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), Multilayer Perceptron (MLP), Boosting and Bagging. The performance metrics observed were Accuracy, Recall, Precision, and F-Score. These metrics, except F-Score, are defined in the equation 9, 10 and 11. The equation 15 defines the latter. Additionally, we obtained confidence intervals for each supervised ML algorithm. Finally, we applied the Kolmogorov-Smirnov test to confirm that the means come from a normal or non-normal distribution, and to know the statistical significance of the difference in mean precisions we applied the Kruskal-Wallis test.

To validate the goodness of the GCN, we test it as follows. The division of the sensors belonging to each cluster was done this way: 80% was used for the training process, 10% was used for the validation phase, and the remaining 10% was used for the testing process. To emulate a cross-validation process, we mix the sensors belonging to each clusters, so 10 different divisions were made to train the network in 10 different ways. We repeat the process for 10 different seeds and look at the precision and confidence intervals. In the same way, we applied the Kolmogorov-Smirnov test to confirm that the means come from a normal or non-normal distribution, and to know the statistical significance of the difference in means of precisions we applied the Kruskal-Wallis test.

$$F - Score(F) = \frac{2 * Pr * Re}{Pr + Re} \quad (15)$$

1.4 Discussion of results

This Section presents the most representative results for each objective proposed in the previous Section. The experimental processes are not described in their entirety since they are already detailed in each of the works related to each of the objectives presented above.

1.4.1 Neural Network Reduction

With the objective of reducing the size of the original architectures and the information necessary to represent their components, we proposed SLRProp, a variant of the Sparse Low Rank method pioneered by [18].

In the experimental process we tested our variant against the original SLR method. There was a noticeable equality in the accuracy obtained by each

Rank k	Cifar10- SLR	Cifar10- SLRProp	Cifar100- SLR	Cifar100- SLRProp	MNIST- SLR	MNIST- SLRProp	Room- SLR	Room- SLRProp
k = 2	17.037%	17.4074%	4%	2.8519%	21.084%	23.492%	17.5%	17.5
k = 4	52.3333%	43.5185%	9.8889%	8.8889%	38.006%	40.053%	17.7%	17.7%
k = 8	89.5926%	81.4444%	47.4444%	37.9259%	71.597%	80.469%	83%	83.9%
$\frac{k}{16} =$	92.963%	92.963%	63.4074%	62.4444%	93.258%	94.245%	-	-
$\frac{k}{32} =$	-	-	-	-	98.416%	98.385%	-	-

Table 1: SLR vs. SLRProp accuracies for different datasets.

method when the number of k was such that the accuracy obtained was identical to that obtained when more columns and rows were kept. There was superiority over the original SLR when both methods were tested on the FC layer prior to the FRL of the VGG16 architecture tested on the Cifar10 dataset. On the contrary, in the pruning applied to the FC layer of VGG16 tested in Cifar100 there was an insignificant difference between both alternatives, and the same occurred in the case of the pruning applied to the FC layer of Lenet5 tested in MNIST and in the case of Room occupancy estimate. However, for the compression regimes where higher data reduction was achieved, the application of SLRProp was superior in terms of accuracy in the case of MNIST, but inferior in the case of the Cifar10 and Cifar100 datasets. All results are detailed in Table 1.

This implies that depending on the nature of the application or task, the best alternative will be different. In some environments where higher compression would be desirable, it would be appropriate to choose a variant that works better for lower k. On the contrary, if the required precision is higher, although the compression rate achieved would not be as high, the variant that obtains better performance metrics would be the one that would be considered. As mentioned above, the compression regime itself does not determine the choice of any of the variants, because these perform differently with the same compression rate when tested on different data sets. In each case, the expert in question would have to determine which option to consider, but the new alternative offers a slight improvement for some scenarios. In general, in more cases the original SLR method was superior to the variant proposed in this work. However, the usefulness of our variant was demonstrated through the experimental process.

1.4.2 Offloading of tasks in the Edge

The best success rate results were obtained when Edge Datacenters were included as possible download destinations. This is an expected consequence of including powerful calculation centers near IoT devices that facilitate transmission to them. These devices have sufficiently powerful processors and large memories to handle the vast majority of the tasks assigned to them. Conversely,

Algorithm	All de- vices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.9194 %	100 %	66.0484 %	99.9194 %	54.5161 %	100 %
Round Robin	99.8387 %	100 %	47.1774 %	99.8387 %	55.4839 %	100 %
GNN	99.9194 %	100 %	80.4032 %	99.9194 %	59.1935 %	100 %
DQN	99.9194 %	100 %	82.0968 %	100 %	65.5645 %	100 %

Table 2: Success rate of different algorithms including different type of destiny devices (10 Edge devices).

Algorithm	All de- vices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.8641 %	100 %	61.8478 %	99.8913 %	97.1739 %	100 %
Round Robin	99.8370 %	99.7011 %	96.30434 %	99.8370 %	59.9185 %	100 %
GNN	99.9185 %	100 %	84.4837 %	99.9185 %	97.1739 %	100 %
DQN	99.9457 %	100 %	84.3478 %	99.9728 %	97.1739 %	100 %

Table 3: Success rate of different algorithms including different type of destiny devices (20 Edge devices).

when tasks were offloaded to the cloud they may not have met latency requirements, and if tasks are offloaded to IoT devices, they may not have been able to respond due to lack of resources. As the number of Edge devices grew, the success rates of cases where Edge datacenters were excluded improved significantly, due to the greater number of free IoT devices. In this case, the number of IoT devices with sufficient processability and memory grew and fewer tasks were transmitted to the cloud. The success rates of different algorithms for 10, 20 and 30 IoT devices are shown in Tables 2, 3 and 4 respectively.

When it comes to the energy efficiency the only noticeable difference was the change in the energy consumption of the IoT devices when the number of IoT devices and consequently the number of tasks grew. This was due to the increase in the number of tasks downloaded to this type of device. The energy consumption of the Edge Datacenters suffered slight increases as the number of IoT devices and consequently the number of tasks grew, but since they were

Algorithm	All de- vices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.9324 %	99.8986 %	70.2196 %	99.8986 %	89.0541 %	100 %
Round Robin	99.1892 %	99.4595 %	91.5541 %	99.8311 %	99.3986 %	100 %
GNN	99.9493 %	100 %	92.3142 %	99.9324 %	93.9696 %	100 %
DQN	99.9662 %	100 %	94.0372 %	99.9662 %	94.5777 %	100 %

Table 4: Success rate of different algorithms including different type of destiny devices (30 Edge devices).

almost complete in all scenarios, the differences were not perceptible compared to those observed in the IoT layer.

The distribution of tasks between different types of devices was favorable for Edge Datacenters, due to their computing capacity and their proximity to the IoT layer. When more tasks were assigned to IoT devices, a higher percentage of them were offloaded to other IoT devices or to the cloud, as a result of the overload situation faced by Edge Datacenters. Figure 10 shows the distribution between different layers.

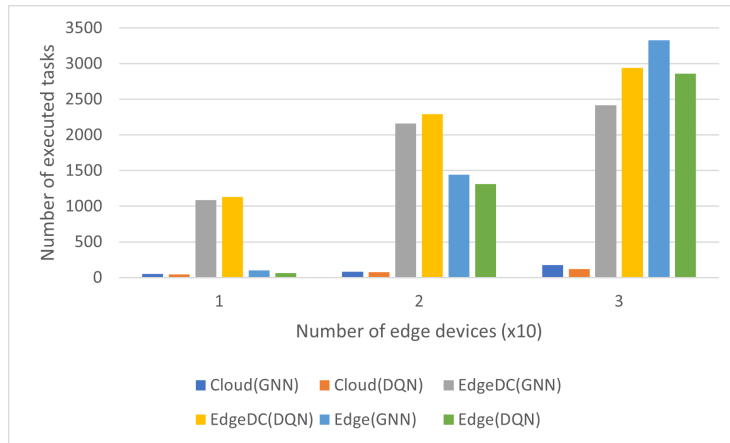


Figure 10: Task distribution with different algorithms and number of Edge devices

The success rates of different layers were different at all. The worst results were obtained by the cloud due to the long time needed to transmit the information through the network to achieve this. Between the Edge devices and Edge Datacenters, the last ones obtained the best success rates. Obviously, the higher capabilities and larger memory were superior in terms of computability compared to the Edge devices, but with a good enough algorithm to orchestrate all the tasks among all possible destinations, offloading the less demanding tasks to weakest computing centers, success rate could be preserved with higher number of generated tasks. A comparative of success rates for different numbers of Edge devices is given in Figure 11 for DQN and GNN algorithms.

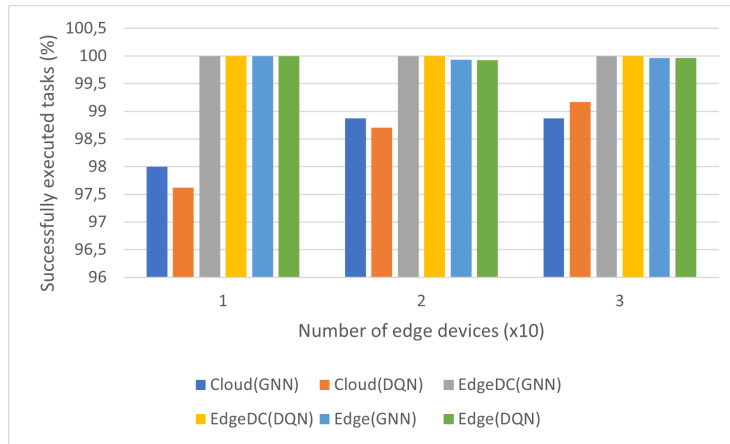


Figure 11: Success rates with different algorithms and number of Edge devices

In almost all possible configurations, GNN and DQN were superior in terms of accuracy and provided good load balancing on the network. Among them there was an insignificant difference in favor of DQN, because DQN decided to offload a greater number of tasks to Edge Datacenters, becoming a better alternative, due to the better performance when Edge Datacenters were included as possible offload destinations. Updated environmental information would benefit the DQN training process, becoming the best alternative to obtain the most precise knowledge about the circumstances of all devices and establish an optimal download policy.

1.4.3 Cooperation of Edge devices

The cooperative application of information provided by sensors provided an interesting insight into the use of buildings and their nature. Clustering methods combined with GNNs proved to be a potential tool to classify different types of buildings using only the mobility patterns of pedestrians entering these establishments.

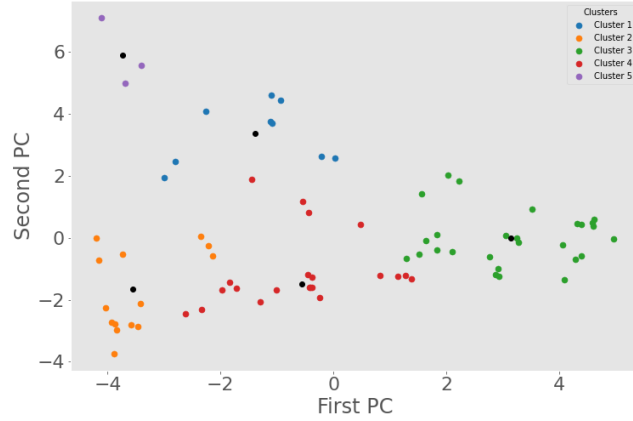


Figure 12: Clustering based on the typology of buildings. Black points indicate the centroids of each cluster.

First, an improvement in the accuracy of the clustering methods was observed after applying dimension reduction techniques. The excess of features was a problem that obstructed the correct division of instances into different clusters. With fewer features used to represent each data, the clustering process achieved a noticeably better result, obtaining more distinguishable groups of instances.

Subsequently, the clusterings of buildings were carried out following different strategies. First, the number of building types surrounding each sensor were the characteristics used to describe each of them. After obtaining the principal components, the K-Means algorithm was applied after evaluating the optimal number of clusters according to the elbow method. After applying dimension reduction, the clusters obtained were 5 using the elbow method. The same process was then repeated for characteristics of mobility patterns across different time intervals of days throughout the week. In this case, 4 clusters reached the point where the graphic flattens out indicating the optimal number of clusters.

Next, the clusterings were carried out knowing the optimal number of centroids. Figure 12 shows the clusters obtained for clustering based on construction characteristics. There are some groups that are better defined than others. For example, Cluster 5 and Cluster 3 are more distinctive than the rest of the Clusters, as instances of different Clusters merge slightly between instances of other Clusters.

Classifier	Accuracy	Precision	Recall	F-Score	Confidence interval
SVM	93.39%	92.99%	93.24%	92.93%	92.64-94.15%
RF	94.14%	93.42%	93.92%	93.08%	93.42-95.07%
DT	88.86%	89.44%	89.05%	88.90%	87.06-90.21%
MLP	81.86%	82.50%	81.80%	81.45%	79.48-84.23%
GNB	91.96%	92.52%	91.89%	91.39%	90.74-93.19%
Adaboost	95.15%	94.28%	93.55%	93.79%	94.41-95.90%
Bagging	91.54%	90.49%	90.59%	90.17%	90.51-92.56%

Table 5: Performance metrics for Building typology dataset based clustering.

To validate these clusters, different supervised ML methods were used. The performance metrics obtained by most of the models used in the process were higher for clustering carried out according to the type of building surrounding the sensor than for clustering based on mobility at different times. The maximum precision was obtained with the Adaboost method with 95.15% and 93.79% F1-Score for the first case. In the latter case, SVM offered the highest performance with 88.89% precision and 88.7% F1-Score. This shows that the typology of the buildings surrounding each of the buildings is a better predictor than the mobility pattern itself. Mobility patterns should predict the essence of each building but this can vary for different reasons. Instead, the typology of the surrounding buildings is a static characteristic, much more reliable than the mobility pattern. All accuracy and F1 score results for the supervised ML models mentioned in Section 1.3.3 are given in Table 5 and 6 respectively .

The correlation between both clusterings was also observed. The general trend followed by the sensors of each cluster was observed in the other data set. We concluded that the sensors that have more activity during the week were mainly related to office buildings, the sensors with more activity on weekends with residential apartments or House/Townhouse type, and those that had similar activity throughout the week were more similar with the Retail type of buildings. This obviously makes sense considering that pedestrians typically spend most of their time at work during the week and at home on weekends. At the same time, the shops are visited throughout the week. The sensors with the most activity at midnight on weekends were of the Townhouse type, which makes sense.

Classifier	Accuracy	Precision	Recall	F-Score	Confidence interval
SVM	87.14%	88.03%	87.32%	87.02%	85.45-88.84%
RF	85.07%	85.03%	84.93%	84.86%	83.66-86.48%
DT	87.43%	87.70%	87.32%	87.02%	85.74-89.12%
MLP	66.71%	66.92%	66.71%	66.31%	64.66-78.66%
GNB	85.71%	88.02%	85.92%	86.40%	83.93-87.50%
Adaboost	81.85%	82.09%	81.78%	81.76%	80.10-83.60%
Bagging	86.10%	86.98%	86.24%	85.83%	84.48-87.71%

Table 6: Performance metrics for time-interval mean activation dataset based clustering.

Finally, considering the grouping carried out following the criterion of sensor activity in different time intervals, GCN was applied, with the characteristics of the nodes being the activities of different time intervals and the characteristics of the edges being the inverse of the distance between the different nodes. The precision obtained was 87%. This shows that the GCN structure offers a slightly better result than clustering itself by observing only pedestrian activity in different time intervals and days of the week. It is worth noting that Table 5 shows the maximum accuracies of the supervised ML algorithms for classifying instances into different groups, being superior for most of the algorithms to the GCN when pedestrian activity was used as a grouping criterion. At the same time, as mentioned above, the typology of most of the buildings was more determining than the mobility patterns of users due to the possible variations that they may suffer. The data collection covered a period of time long enough to reflect the real trend of pedestrians, not being a problem and hampering the predictive strength of this characteristic. Table 5 reviews the accuracies obtained following the 3 ways presented in this subsection in comparison with the results obtained by [38].

1.5 Relevance of Results

In this Section we state the contribution and impacts of works that compound this Thesis to the literature. The chronological order of the publications of the works does not follow the order in which the objectives have been presented in this Thesis report. This is not the order in which the works have been published:

Method	Average Accuracy
Top-1 Clustering(Activity)	88.89%
Top-1 Clustering(Building Typology)	95.15%
Top-1 GCN(Activity)	87.00 %
L. Zhuo et al.[38]	85.66%

Table 7: Comparative of results with the literature.

- The first contribution on *Reviewing and Discussing Graph Reduction in Edge Computing Context* was presented in International Conference on Statistics and Machine Learning in Electronics (ICSMLE 2022) and published as conference paper in "MDPI Computation"[39]. We committed with an lengthened version in Sensors journal in 2023 under the title *SLR-Prop: A Back-Propagation Variant of Sparse Low Rank Method for DNNs Reduction* [4].

Journal Ranking:

MDPI Sensors (2023). Category: Chemistry, Analytical; Engineering, Electrical & Electronic; Instruments & Instrumentation.

2022 Journal Impact Factor: 3.9 (100/275 Q2). JIF Percentile: 63.8.

- The second contribution on *Task Offloading in Edge Computing using GNNs and DQN* was published in journal Computer Modelling in Engineering & Sciences in 2023.

Journal Ranking:

Computer Modeling in Engineering & Sciences (2023). Category: Mathematics.

2022 Journal Impact Factor: 2.4 (34/107 Q2). JIF Percentile: 68.7. -

- Finally, we contributed publishing the work named *Graph based learning for building prediction in Smart Cities* in journal IEEE Access in 2022.

Journal Ranking:

IEEE Access (2022). Category: Multidisciplinary.

2022 Journal Impact Factor: 3.9 (73/158 Q2). JIF Percentile: 54.1.

- Independently, we contributed in the 18th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2023), with the work named *Comprehensive Analysis of Different Techniques for Data Augmentation and Proposal of New Variants of BOSME and GAN* and was published as conference paper in "Hybrid Artificial Intelligent Systems" [40].

Chapter 2

Conclusions & Discussion

In this thesis we have made different contributions that can be divided into 2 groups. Some of them have focused on performing complex ML tasks at and near the IoT layer, offering ways to reduce models and orchestrate load balancing between IoT and Edge layer. The other part of the work has focused on applications based on the data obtained and stored by IoT devices.

Throughout the thesis we have covered the 2 general objectives presented in Section 1.3. The algorithmic/computation objective have been studied and developed in the three works that made up this thesis, specially being the core of the [4][5]. The modification and computation of complex DL models have been carried out in an efficient way, obtaining promising results in term of optimisation of memory usage, load balancing and performance. The second general objective, which is related to the application of these models, was demonstrated in [6] as a valuable and novel source of knowledge. The specific objectives presented in Section 1.5 were met, i.e., the reduction of DL models to adapt them to resource-constrained devices, the offloading of resource-intensive tasks from IoT devices to the Edge layer along with the correct orchestration of tasks between different devices, and finally applications for Smart Cities using the data acquired by these IoT devices and stored for a long period of time in remote servers or Datacenters. Each objective has been structured around a topic. In case of the reduction of models the main topic has been Pruning techniques, Edge Offloading has been the second topic of interest and finally Building classification has been the research interest of the last specific objective.

In each case, the methodology used has been the following. First, in order to obtain a general overview of the framework of the issue in question an intensive search of the actual and most relevant works has been done. Later, the weaknesses or the gaps that could be filled have been identified. Finally, the solutions to these have been proposed with the alternatives to the algorithms/policies or novel applications, and these have been tested against the most recent relevant works.

According to the first specific objective, the introduction of the concept of backpropagation of relevances from the FRL to the rest of the layers of the original network does not always overcome the assumption of relevances independently within the layers. However, the general result shows that the SLRProp method is slightly better than the original version of sparsification presented by [18]. In this way, the breakthrough presented by [37] is preserved in this experiment; relevances propagated between different layers through connections between neurons. This shows that the components of each layer have certain influence on the rest of the network components, even though the main contribution to the final result of each component is more connected with other aspects than the absolute values of the weights of the connections across layers. In this case, the cost defined as the difference of the accuracy between the case when a certain component is eliminated from the original network and the original structure’s accuracy showed that it might be more crucial when selecting which connections should be removed when pruning the original network.

The proposed alternative offers a slight improvement in different accuracy metrics, but is still too costly in terms of time efficiency and computational load. Considering the absolute values of the weights of these networks as a criteria to decide which columns and rows are maintained in the sparsified matrices offers a near identical result in terms of accuracy that needs $\sim 100x$ less time for sparsifying the SVD matrices in the training phase. In applications where time response is crucial, this last alternative method may be more adequate.

According to the second specific objective, our proposed algorithms outperform the default PureEdgeSim simulator methods in terms of success rate and load balancing. In other works such as [41] they achieved an average improvement of 20.48%, 16.28% and 12.36% with respect to random download, higher data rate downloading (HDR) and the highest computing device (HCD) respectively. In [42] they achieved a 20% reduction in total computation delay and 25% reduction in average computation delay compared to the GK-means DQN-based offloading policy. In our case, the rest of the offloading policies analyzed offered a better result, since they offered decent behavior in most cases. However, our methods significantly improved the success rates of the mentioned algorithms that offer quite similar energy consumption and this has more to do with the distribution of tasks in different layers. Our network environments and experimental setup are completely different compared to those used in the works just mentioned. Therefore, the comparison cannot be made directly between different works.

The distribution of tasks was different in our two algorithms, with a larger number of tasks being offloaded to the Edge Datacenters when DQN was applied. This resulted in a slight improvement in the success rate due to the greater capabilities of this type of computing centers. Between both types of algorithms, the best results were offered by DQN with a slight variation. The

ability to obtain the optimal policy increased when the number of Edge devices and, consequently, the number of generated tasks was larger. The same was true for GNN: by having more nodes and a broader network structure, the algorithm was able to reach a near-optimal offloading decision.

According to the third specific objective, the potential of clustering methods, specifically K-means algorithm, at identifying groups of sensors following different criteria has been demonstrated. The typology of the rest of the buildings surrounding the buildings themselves was the main factor when dividing sensors into different groups. Regarding the clustering made following the activity patterns, the overall activity of the pedestrians in these type of buildings does not determine in a clear way their typology. The datasets used and the length of time period in which the data were detected and stored were sufficient, but in other experimental setups the results may probably be a little closer to the first case of clustering.

As regards GCN, the results were better for the same grouping criteria (activity patterns). The graph, thanks to its architecture, eased the learning of the structure of the city composed of buildings. The edges' feature (inverse of distance) was a determining characteristic of the buildings, due to the similarity between buildings that are close to each other.

Limitations

Throughout the different researches that have been completed during the thesis, some limitations have been found in terms of computability and data availability.

- The training and validation of different algorithms have been designed regarding the time availability and the processing and memory resources. The number of iterations and different seeds were fixed so as not to exceed the predicted computation time.
- The datasets used for training and testing different models were selected according to the predicted training time in each case. However, the datasets used were sufficient to verify the goodness and quality of the proposed alternatives, solving the problems or deficiencies found in the literature.

Future work

From the works presented, new lines of research emerge that are proposed as future work. Due to the environmental dilemma that we are globally facing these days, there is a special interest on reducing the carbon footprint and minimizing energy usage. Thus, the objective for the future research work is the reduction of energy usage in these type of techniques and applications. Green algorithms are becoming more diverse and their popularity is increasing in an

unprecedented way.

According to the first work, the reduction of the models will also be enhanced by achieving a variant that deals with the minimization of energy consumption. Reducing the inference and training time of these reduction techniques would be a key factor to minimize the computation time and consequently the energy consumption of these operations.

According to the second work, the workload balance between the Edge and IoT layer guarantees the completion of the tasks within the required latency. However, the Edge Datacenters would have a higher energy consumption than the IoT devices. Consequently, the optimization of the offloading policy would take care of the total energy consumption of the entire network and consider this last at dividing workload among different devices while complying latency requirements.

According to the third work, knowledge of the building typology would facilitate the optimal supply of energy and lighting in these urban areas, avoiding energy waste in less busy buildings at certain time intervals throughout the day on different days of the week. The inclusion of more than one criterion to group buildings may improve the results obtained in this thesis, including determining factors such as those we have treated independently.

Bibliography

- [1] Firdose Saeik, Marios Avgeris, Dimitrios Spatharakis, Nina Santi, Dimitrios Dechouniotis, John Violos, Aris Leivadreas, Nikolaos Athanasopoulos, Nathalie Mitton, and Symeon Papavassiliou. Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions. *Computer Networks*, 195:108177, 2021.
- [2] Arthur L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers. I*, pages 335–365. Springer New York, New York, NY, 1988.
- [3] Inneke Mayachita. *Understanding Graph Convolutional Networks for Node Classification*. <https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b>.
- [4] Asier Garmendia-Orbegozo, Jose David Nuñez-Gonzalez, and Miguel Angel Anton. Slrprop: A back-propagation variant of sparse low rank method for dnns reduction. *Sensors*, 23(5), 2023.
- [5] Miguel Angel Anton Asier Garmendia-Orbegozo, Jose David Nunez-Gonzalez. Task offloading in edge computing using gnns and dqn. *Computer Modeling in Engineering & Sciences*, 2024.
- [6] Asier Garmendia-Orbegozo, Sarah Noye, Miguel Angel Anton, and J. David Nuñez-Gonzalez. Graph based learning for building prediction in smart cities. *IEEE Access*, 10:45471–45484, 2022.
- [7] Mithilesh Shirsat. *The Intersection of Machine Learning and IoT*. [Post]. LinkedIn. <https://www.linkedin.com/pulse/intersection-machine-learning-iot-mithilesh-shirsat/>.
- [8] James McCaffrey. Machine learning with iot devices on the edge. *MSDN Magazine Issues*, 33(7), 2018.
- [9] Shahid Anwar, Jasni Mohamad Zain, Mohamad Zolkipli, Zakira Inayat, Suleman Khan, Bokolo Anthony Jnr, and Victor Chang. From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions. *Algorithms*, 2017, 03 2017.

- [10] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.
- [11] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993.
- [12] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. 11 2016.
- [13] Chaohui Yu, Jindong Wang, Yiqiang Chen, and Zijing Wu. Transfer channel pruning for compressing deep domain adaptation models. In Leong Hou U. and Hady W. Lauw, editors, *Trends and Applications in Knowledge Discovery and Data Mining*, pages 257–273, Cham, 2019. Springer International Publishing.
- [14] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.
- [15] R Muthukrishnan and R Rohini. Lasso: A feature selection technique in predictive modeling for machine learning. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, pages 18–20, 2016.
- [16] Maksim Kholiavchenko. Iterative low-rank approximation for cnn compression. 03 2018.
- [17] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 67–76, 2017.
- [18] Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, 02 2020.
- [19] Analytics india magazine — artificial intelligence, data science, machine learning. <https://analyticsindiamag.com/>.
- [20] Sima Jeddi and Saeed Sharifian. A water cycle optimized wavelet neural network algorithm for demand prediction in cloud computing. *Cluster Computing*, 22, 12 2019.
- [21] Seung-Seob Lee and Sukyoung Lee. Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. *IEEE Internet of Things Journal*, 7(10):10450–10464, 2020.

- [22] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.
- [23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015.
- [25] Muhammad Islam, Abdulsalam S. Dukyil, Saleh Alyahya, and Shabana Habib. An iot enable anomaly detection system for smart city surveillance. *Sensors*, 23(4), 2023.
- [26] Charafeddine Mechalikh, Hajer Taktak, and Faouzi Moussa. Pureedgesim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments. In *2019 International Conference on High Performance Computing Simulation (HPCS)*, pages 700–707, 2019.
- [27] City of melbourne open data. <https://data.melbourne.vic.gov.au/pages/home/>.
- [28] City of madrid open data. <https://datos.madrid.es/portal/site/egob>.
- [29] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.
- [30] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.
- [31] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane,

- Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
- [32] Francois Chollet et al. Keras, 2015. <https://github.com/fchollet/keras>.
- [33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [34] Kolby Nottingham Markelle Kelly, Rachel Longjohn. *The UCI Machine Learning Repository*. <https://archive.ics.uci.edu>.
- [35] Built In: National Tech Startups. Fully connected layer vs. convolutional layer: Explained. <https://builtin.com/machine-learning/fully-connected-layer>, 2022. [Online; accessed 26-October-2023].
- [36] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.
- [37] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. Nisp: Pruning networks using neuron importance score propagation, 2017.
- [38] Li Zhuo, Qingli Shi, Chenyang Zhang, Qiuping Li, and Haiyan Tao. Identifying building functions from the spatiotemporal population density and the interactions of people among buildings. *ISPRS International Journal of Geo-Information*, 8(6), 2019.
- [39] Asier Garmendia-Orbegozo, José David Núñez-Gonzalez, and Miguel Ángel Antón. Reviewing and discussing graph reduction in edge computing context. *Computation*, 10(9), 2022.
- [40] Asier Garmendia-Orbegozo, Jose David Nuñez-Gonzalez, Miguel Angel Anton Gonzalez, and Manuel Graña. Comprehensive analysis of different techniques for data augmentation and proposal of new variants of bosme and gan. In Pablo García Bringas, Hilde Pérez García, Francisco Javier Martínez de Pisón, Francisco Martínez Álvarez, Alicia Troncoso Lora, Álvaro Herrero, José Luis Calvo Rolle, Héctor Quintián, and Emilio Corchado, editors, *Hybrid Artificial Intelligent Systems*, pages 145–155, Cham, 2023. Springer Nature Switzerland.
- [41] Mohammed Maray, Ehaz Mustafa, Junaid Shuja, and Muhammad Bilal. Dependent task offloading with deadline-aware scheduling in mobile edge networks. *Internet of Things*, 23:100868, 2023.

- [42] Liang Zhao, Zijia Zhao, Enchao Zhang, Ammar Hawbani, Ahmed Al-Dubai, and Amir Hussain. A digital twin-assisted intelligent partial offloading approach for vehicular edge computing. *IEEE Journal on Selected Areas in Communications*, 08 2023.




Appendix A

Articles

- A.1 **SLRProp: A Back-Propagation Variant of Sparse Low Rank Method for DNNs Reduction**

Article

SLRProp: A Back-Propagation Variant of Sparse Low Rank Method for DNNs Reduction

Asier Garmendia-Orbegozo ^{1,†} , Jose David Nuñez-Gonzalez ^{1,*,†}  and Miguel Angel Anton ^{2,†} ¹ Department of Applied Mathematics, University of the Basque Country UPV/EHU, 20600 Eibar, Spain² TECNALIA, Basque Research and Technology Alliance (BRTA), 20009 San Sebastian, Spain

* Correspondence: josedavid.nunez@ehu.eus

† These authors contributed equally to this work.

Abstract: Application of deep neural networks (DNN) in edge computing has emerged as a consequence of the need of real time and distributed response of different devices in a large number of scenarios. To this end, shredding these original structures is urgent due to the high number of parameters needed to represent them. As a consequence, the most representative components of different layers are kept in order to maintain the network's accuracy as close as possible to the entire network's ones. To do so, two different approaches have been developed in this work. First, the Sparse Low Rank Method (SLR) has been applied to two different Fully Connected (FC) layers to watch their effect on the final response, and the method has been applied to the latest of these layers as a duplicate. On the contrary, SLRProp has been proposed as a variant case, where the relevances of the previous FC layer's components were weighed as the sum of the products of each of these neurons' absolute values and the relevances of the neurons from the last FC layer that are connected with the neurons from the previous FC layer. Thus, the relationship of relevances across layer was considered. Experiments have been carried out in well-known architectures to conclude whether the relevances throughout layers have less effect on the final response of the network than the independent relevances intra-layer.

Keywords: pruning; deep learning; edge computing

Citation: Garmendia-Orbegozo, A.; Nuñez-Gonzalez, J.D.; Anton, M.A. SLRProp: A Back-Propagation Variant of Sparse Low Rank Method for DNNs Reduction. *Sensors* **2023**, *23*, 2718. <https://doi.org/10.3390/s23052718>

Academic Editor: Juan M. Corchado

Received: 19 January 2023

Revised: 16 February 2023

Accepted: 28 February 2023

Published: 2 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The use of deep neural networks (DNN) in different scenarios related to Machine Learning (ML) applications has developed in such a way that currently neural network designs have billions of parameters with a great capability of prediction, as one of the most used types of architecture in prediction tasks. Specifically, some of those applications include image, sound, and textual data recognition. In contrast to other ML algorithms, the DNNs have achieved a remarkable accuracy. However, the use of these networks in memory and processing resource constrained devices is limited due to the amount of data needed to develop these architectures and the high computation costs for training them. Consequently, different reduction techniques are essential to fit these former networks in resource constrained devices, such as edge devices.

Among others, the most used and effective way to shrink these networks is the use of techniques such as pruning and quantization. The former one consists of removing parameters (neurons or weights) that have negligible contribution while maintaining the accuracy of the classifier. On the other hand, quantization involves replacing datatypes to reduced width datatypes, by transforming data to fit into new datatypes' shapes. In this way, reduced networks are able to compete with the original ones in terms of accuracy and even improve these in some cases in which overfitting issues were hindering their predictability. Moreover, by reducing the width of the data, edge devices could face the storage issue mentioned above and collect larger datasets in constrained memory sizes.

Mainly convolutional neural networks (CNN) became a widely used network structure in image recognition tasks. Such a success is built upon a large number of model parameters and convolutional operations. As a result, the huge storage and computation costs make these models difficult to be deployed on resource-constrained devices, such as phones and robots, needing to adopt different reduction techniques.

In this work, we introduce a new variant to the Sparse Low Rank (SLR) method to develop weight pruning in well-known architectures, SLRProp. We judge that the last Fully Connected (FC) Layer, Final Response Layer (FRL), is the most relevant to the final decision. Moreover, the relevance of weights of this final layer are propagated to the previous layers, making each neuron non-independent of the previous layers in terms of relevance. Consequently, the connections of each neuron has a direct relationship with neuron's predictability in the final decision of the network, needing to consider them. After factorizing the weight matrices of FC Layers, we sparsified them only considering the most relevant parts and propagate these relevances to the previous FC layers by considering the connections between different FC layers. Similarly, we performed a parallel process in which the sparsification of matrices has been carried out independently between layers, only considering the relevance intra-layer. Finally, we state the validity of the supposition of backpropagating the relevance within layers. As a result, the pruning process is optimized by determining the less relevant components of each layer, as a consequence of the addition of the backpropagation concept to the Sparse Low Rank Method contributed in this work.

State of the Art

There have been several attempts to reduce DNNs dimensionality by applying the techniques mentioned above. Pruning techniques consist of removing part of connections (weights) or neurons from the original network so as to reduce the dimension of the original structure by maintaining its ability to predict. The core of these techniques reside on the redundancy that some elements add to the entire architecture. Memory size and bandwidth reduction are addressed with these techniques. Redundancy is lowered and overfitting is faced in some scenarios. Different classifications of works based on this ability are made depending on element pruned, structured/unstructured (symmetry), and static/dynamic.

Static pruning is the process of removing elements of a network structure offline before training and inference processes. During these last processes no changes are made to the network previously modified. However, removal of different components of the architecture requires a fine-tuning or retraining of the pruned network. This is a direct consequence of the changes that suffer the network by removing big part of its elements. Thus, some computation effort is needed in order to reach comparable accuracy to the original network.

The pruning has been carried out by following different criteria. In [1,2], they used the second derivative of the Hessian matrix to reduce the dimension of the original architecture. Optimal Brain Damage (OBD) and Optimal Brain Surgeon (OBS) work under three assumptions. Quadratic: the cost function is near quadratic. Extremal: the pruning is conducted after the network converged. Diagonal: sums up the error of individual weights by pruning the result of the error caused by their co-consequence. Additionally, OBS avoids the diagonal assumption and improves neuron removal precision by up to 90% reduction in weights for XOR networks. Using Taylor expansions of first order [3,4] was also an alternative to the previous ones to tackle networks' dimension issues, as a criterion to approximate the change of loss in the objective function as an effect of pruning.

Some works were based on the magnitude of the elements themselves. It is undoubtedly true that near-zero values of weights make far less contributions to the results than others that surpass a certain threshold value. In this way, removing connections that may appear unnecessary, the original network is shrunk. The best way to accomplish this is the removal of weights layer by layer to not abruptly decrease the performance of the resulting network. LASSO [5] was introduced as a penalty term. It shrinks the least absolute valued feature's corresponding weights by increasing weight sparsity. This operation has been

shown to offer a better performance than traditional procedures such as OBS by selecting the most significantly contributed variables instead of using all the variables, achieving approximately 60% more sparsity than OBS. The problem with LASSO is that it is an element-wise pruning technique leading to an unstructured network and sparse weight matrices. By performing this technique dividing the process by groups—as Group LASSO [6] does, removing entire groups of neurons and maintaining the original network’s structure—this last issue was solved. Groups are made based on geometry, computational complexity, or group sparsity, among others.

Singular Value Decomposition (SVD) is an effective and promising technique to shred convolutional or FC layers by reducing the number of parameters needed to represent them. Not only it has been useful for image classification tasks, but also in object detection [7] scenarios and others related with DNN-based acoustic modeling [8,9]. Low-rank decomposition for convolution layers as well as fully connected layers were applied in several works. Kholiavchenko et al. [10] proposed an iterative approach to low-rank decomposition by applying dynamic rank selection to image classification and object detection models. One of its negative aspects was that iteratively applying low-rank decomposition needs longer time and higher computational resources for rank selection in deeper models. The alternative proposed by [11] assumes the properties of both low-rank and sparseness of weight matrices while aiming to reconstruct the original matrix. In [12], through mixing the concepts of sparsity and existence of unequal contributions of neurons towards achieving the target, the Sparse Low Rank (SLR) method is proposed—a method that scatters SVD matrices to compress them by conserving lower rank for unimportant neurons. As a result, it is feasible to reduce the 3.6× storage space of SVD without much variance on the model accuracy. Speedup in the computation was another advantage that has the structured sparsity obtained by the presented approach.

The majority of the previous works had paid attention to the individual pruning of layers while not considering the connection between different layers. In [13], they claimed that the last FC layer is the most relevant of the entire network regarding the effect on the final response of the entire network. Considering this last, they proposed to prune the previous layer of the network when considering the connections of neurons with the neurons of this last FC layer called the Final Response Layer (FRL). In this way, the relevances of the neurons considered independently for the FRL were backpropagated to the previous layer’s neurons. The pruning of the rest of the layers was carried out similarly, scoring the relevance of the neurons when considering the connections with the posterior layers’ neurons.

Other alternatives have been proposed to carry out static pruning. In [14], they proposed an innovative method for CNNs pruning called layerwise relevance propagation. Each unit’s relevance to the final decision is measured, and the units that are below a predefined threshold are removed from the original structure. As a last step, each component’s relevance is recalculated by calculating the total relevance per layer to keep it constant through the iterations. Thus, each unit’s relevance is recalculated to maintain this value. In [15], a method of pruning redundant features along with their related feature maps, according to their relative cosine distances in the feature space, is proposed, and the authors achieve smaller networks with a significant download in post-training inference computational costs and achieving a decent performance. Redundancy can be minimized while inference cost (FLOPS) is reduced by 40% for VGG-16, 28%/39% for ResNet-56/110 models trained on CIFAR-10, and 28% for ResNet-34 trained on ImageNet database with almost negligible loss of accuracy. To fix the decrease in accuracy after pruning, models were retrained for some iterations maintaining all hyper-parameters.

2. Material and Methods

In this section, we describe the methodology proposed in order to attempt to improve the results obtained in the literature for different neural networks and datasets. Additionally, we present the datasets and models used for experimentation.

2.1. Methodology

The approach we present in this study follows this methodology. First, traditional low-rank decomposition SVD is applied to the weight matrix of the final FC layer, called FRL. Next, input and output weights in the layer are selected for sparsification using different neuron selection strategies. Then, sparsification is applied to the selected input and output neuron components in the decomposed matrices. With the most relevant neurons of the final FC layer obtained we back propagate their relevance to the prior FC layer, following the idea proposed by [13], and we obtained the relevance of the neurons composing the prior FC layer. Finally, we repeated the process of sparsification for the decomposed matrices of the prior FC layer. In parallel, we performed the same process of sparsification but only considering the relevance of each individual layer for the last two FC layers. The results and comparative of both methodologies are summarized in Section 4.

2.1.1. Single Value Decomposition (SVD)

One way for decomposing matrices representing the weights of neural networks is the use of low-rank factorization. A convolutional neural network is composed of a large number of convolutional layers and fully connected layers. By applying this technique to convolutional kernels weights optimization of the inference speed, the convolution operation could be obtained due to the reduction in the time needed for multiplication with factorized matrices compared to that of multiplication with 3D weights of kernels.

In a FC layer having m input and n output neurons, activation $a \in \mathbb{R}^n$ of the layer with n nodes is represented as

$$a = \mathbf{g}(\mathbf{W}^T \mathbf{X} + \mathbf{b}) \quad (1)$$

where \mathbf{X} represents the input to the layer, and $\mathbf{g}()$ represents any of the possible activation functions. FC layers connections form a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ and a bias vector $\mathbf{b} \in \mathbb{R}^n$ where each parameter in the weight matrix \mathbf{W} is $w^{ij} \in \mathbb{R}$ ($1 \leq i \leq m, 1 \leq j \leq n$), and bias matrix \mathbf{b} is $b^j \in \mathbb{R}$ ($1 \leq j \leq n$). The proposed approach is applied to the weight matrix \mathbf{W} after training the entire model. The SVD approach decomposes the weight matrix \mathbf{W} as $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{V}^T \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\mathbf{S} \in \mathbb{R}^{m \times n}$ is a diagonal matrix.

2.1.2. Sparse Low Rank Decomposition

The matrix \mathbf{S} is a diagonal matrix containing n non-negative singular values in a decreasing order. The k singular values that are most significant are kept by Truncated SVD where the decomposed matrices \mathbf{U} , \mathbf{S} , and \mathbf{V}^T become $\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}^T \in \mathbb{R}^{m \times k}, \mathbb{R}^{k \times k}, \mathbb{R}^{k \times n}$. By this way, the original weights \mathbf{W} are replaced into reconstructed approximated weight $\hat{\mathbf{W}}$ as $\hat{\mathbf{W}} = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T$.

In SVD we have diagonal matrix sigma \mathbf{S} with the most significant singular values from the upper left to lower right in a decreasing order. In the truncation process the first k rows of \mathbf{U} and columns of $\hat{\mathbf{V}}^T$ are kept.

Simulating the approach driven by [12] we compressed truncated matrices $\hat{\mathbf{U}}, \hat{\mathbf{S}}$, and $\hat{\mathbf{V}}$ based on the importance of the m input and n output neurons, i.e., we represented a few columns of $\hat{\mathbf{U}}$ and rows of $\hat{\mathbf{V}}^T$ with a rank lower than k , called reduced rank rk . In this way, only rk most significant rows and columns are kept in $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}^T$, respectively, due to the order of importance of \mathbf{W} that starts from left to right through columns of $\hat{\mathbf{U}}$ and top to bottom through rows of $\hat{\mathbf{V}}^T$. We considered only the most significant rows (rm) and columns (rn) from each column and row from $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}^T$, respectively, following the cost criteria, briefly explained in the next subsection.

When the matrices $\hat{\mathbf{U}}, \hat{\mathbf{S}}$ and $\hat{\mathbf{V}}^T$ are sparsified with sr and rr , the total number of non-zero parameters of the $\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}^T$ become $k(m - rm + n - rn + 1) + rk(rm + rn)$, which is less than the number of non-zero parameters of truncated SVD $k(m + n + 1)$.

Pruning fully connected layers is much more effective in terms of accuracy, time, and energy efficiency than pruning convolutional layers as shown in [16], which contributes to bigger losses in prediction capability with the same rate of reduction in parameters. Those are usually placed in the first positions in DNNs, and they are more sensitive than the ones that are placed in the last positions in many cases. In this study, we followed the approach directed by [12] sparsifying SVD matrices achieving a low compression rate without big losses in accuracy. We used as a metric of sparsification the compression rate defined in [12], as the ratio between the parameters needed to define the sparsified decomposed matrices and the original weights' matrix parameters. In our case, we analyzed their 3 variants of applying SLR, that were based in cost, weights, and activations, and we proposed two new variants that sum the importance of cost and weights and cost and activations due to the fact that each of them performed as the best variant in different compression rate regimes.

Overall, the most relevant attribute was the cost, so we decided to establish this as the criteria for selection of the rows and columns for sparsification. An explanation of the full process of this method is given in Algorithm 1.

Algorithm 1 SLRProp

```

Weights1 ← FRL weights
Weights2 ← Previous FC layer weights
U, S, V ← SVD(Weights1)
tU, tS, tV ← U[:, 0 : rank], S[0 : rank], V[0 : rank, :]
for Nrows do
  tempU[row, rk :] = 0
  Weightst ← tempU * tS * tV
  Score ← Accuracy(Weightst)
  is ← Ranking of rows
end for
for Ncolumns do
  tempV[rk :, column] = 0
  Weightst ← tU * tS * tempV
  Score ← Accuracy(Weightst)
  os ← Ranking of columns
end for
U(rmrows) ← 0 where rm=sr*m
V(rncolumns) ← 0 where rn=sr*n
U2, S2, V2 ← SVD(Weights2)
tU2, tS2, tV2 ← U2[:, 0 : rank], S2[0 : rank], V2[0 : rank, :]
for Nrows do
  Score ←  $\sum Abs(U2[i, j] * is[j])$ 
  is2 ← Ranking of rows
end for
for Ncolumns do
  tempV2[rr :, column] = 0
  Weightst2 ← tU2 * tS2 * tempV2
  Score ← Accuracy(Weightst2)
  os2 ← Ranking of columns
end for
U2(rmrows) ← 0
V2(rncolumns) ← 0

```

2.1.3. Selection of Rows and Columns Based on Cost

A neuron's importance is defined by whether there is a change or not in the network performance after removing it. Let c be the default cost of the neural network with original trained weight W estimated for the p training samples, computed using any loss function. Let \hat{c} be the value of cost of the network with sparsified weights \hat{W} . By truncating with

reduced rank rr a specific row of $\hat{\mathbf{U}}$ or column of $\hat{\mathbf{V}}^T$ we have the absolute change in cost is or os. Those are calculated as follows:

$$is_i = |c - \hat{c}_i| \quad (2)$$

$$os_j = |c - \hat{c}_j| \quad (3)$$

As the sparsification process purpose is to ensure that the functionality of the network does not change after compression, and not to reduce the overall network cost or improve accuracy, only the absolute change in the cost value is considered.

2.1.4. Propagation of Relevance between Layers

As it is known, the majority of neural networks can be formulated as a nested function. Thus, we can define a network with n hidden layers as a $F^{(n)} = f^{(n)} \circ f^{(n-1)} \circ \dots \circ f^{(1)}$. Each layer can be represented as follows:

$$f^{(n)}(x) = \sigma^{(n)}(\mathbf{w}^{(n)}x + \mathbf{b}^{(n)}) \quad (4)$$

where $\sigma^{(n)}$ is the activation function of each layer, $\mathbf{w}^{(n)}$ is the corresponding layers connections' weight function, and $\mathbf{b}^{(n)}$ is the bias of each layer. At this stage it is possible to say that all of these layers are interconnected and each of them has direct relevance on the final decision of the entire network. Consequently, weights from the FRL, that is the last Fully Connected Layer, backpropagate their relevance to the prior layers as proposed in [13]. As a result, the relevance of each neuron in the final decision is the composition of weights that are interconnected until the FRL corresponding element's relevance. The summation of the corresponding relevances is given by Equation (5).

$$s_k = |\mathbf{w}^{(k+1)}|^T |\mathbf{w}^{(k+2)}|^T \dots |\mathbf{w}^{(n)}|^T s_n \quad (5)$$

The absolute value of the weights that are connected to each of the neurons of the FRL are multiplied by the relevance of these in the FRL.

$$s_{k,j} = \sum_i |w_{i,j}^{(k+1)}| s_{k+1,i} \quad (6)$$

Equation (6) shows the relevance of the j -th neuron in the k -th layer, which propagates the relevances of the neurons from the posterior $k + 1$ -th layer that are connected with it.

By introducing this idea to the SVD matrices, keeping only the most relevant rows of \mathbf{U} matrices, we can consider only the most relevant neurons of that layer. The procedure in the FC layers that are not the FRL, is similar to the original SLR method except for the sparsification of the \mathbf{U} matrices where the relevance propagated through the posterior layers is considered to determine the most relevant neurons. This relevance is propagated following Equation (6).

In summary, the main contributions made by this work are the following. The pruning of weights carried out in these FC layers is more optimal than in the original SLR method. Consequently, the performance of the resulting network is raised, obtaining sub-optimal results in terms of different performance metrics defined in Section 4 with far less weights needed compared with the original structure. Thus, in scenarios in which original network structures cannot fit end user devices due to memory restrictions are crucial for such reduction techniques.

2.2. Materials

Regarding the materials, we used two well-known models for image recognition, VGG-16 [17] and Lenet5 [18], where VGG architecture is much known for its memory intensive FC layers. It is worth noting that VGG is the commonly used architecture with FC layers where other popular image recognition models, such as ResNet, Inception, MobileNet,

ResNet, DenseNet, and object detection models, do not have FC layers except the final softmax layer. Tables 1 and 2 show the specifications of each network structure.

These two different approaches were tested on different well-known datasets, Cifar10 (VGG16), Cifar100 (VGG16), and MNIST (Lenet5). Each of them contain 32×32 images (color images in Cifar10/Cifar100 and grayscale images in MNIST). In case of Cifar10 and MNIST there are 10 different classes and 100 in Cifar100. All of them have been trained using default 10,000 test images and 50,000 and 60,000 training images for the Cifar and MNIST datasets, respectively. Different compression rates were applied for sparsifying SVD matrices; therefore, for each dataset we obtained different performance metrics for each method. Overall, we were able to state which method was the best in each case. The datasets used for experiments comprise a good mix of different image types, sizes, and number of classes. CIFAR-10 and CIFAR-100 have general purpose image classes where MNIST dataset contains handwritten digit images.

Moreover, to demonstrate the usefulness of our approach in sensor related data we tested our approach in a model consisting of 3 FC layers for the Room Occupancy Estimation Data Set from the UCI Machine Learning Repository. It is a dataset for estimating the precise number of occupants in a room using multiple non-intrusive environmental sensors such as temperature, light, sound, CO₂, and PIR. There are 10,129 instances using 1000 for testing and the rest for training. Table 3 shows the specifications of the network structure.

Table 1. VGG16 model trained for 32×32 images.

Layer Name	Layer Type	Feature Map	Output Size of Images	Kernel Size	Stride	Activation
Input	Image	1	$32 \times 32 \times 3$	-	-	-
Conv-1	$2 \times$ Conv	64	$32 \times 32 \times 64$	3×3	1	relu
Pool1	Maxpool	64	$16 \times 16 \times 64$	3×3	2	relu
Conv-2	$2 \times$ Conv	128	$16 \times 16 \times 128$	3×3	1	relu
Pool2	Maxpool	128	$8 \times 8 \times 128$	3×3	2	relu
Conv-3	$2 \times$ Conv	256	$8 \times 8 \times 256$	3×3	1	relu
Pool3	Maxpool	256	$4 \times 4 \times 256$	3×3	2	relu
Conv-4	$2 \times$ Conv	512	$4 \times 4 \times 512$	3×3	1	relu
Pool4	Maxpool	512	$2 \times 2 \times 512$	3×3	2	relu
Conv-5	$2 \times$ Conv	512	$2 \times 2 \times 512$	3×3	1	relu
Pool5	Maxpool	512	$1 \times 1 \times 512$	3×3	2	relu
Flatten	Flatten	-	512	-	-	relu
FC6	Dense	-	4096	-	-	relu
FC7	Dense	-	4096	-	-	relu
FC8	Dense	-	# of classes	-	-	softmax

Table 2. Lenet5 model trained for 32×32 images.

Layer Name	Layer Type	Feature Map	Output Size of Images	Kernel Size	Stride	Activation
Input	Image	1	$32 \times 32 \times 3$	-	-	-
Conv-1	$1 \times$ Conv	6	$28 \times 28 \times 6$	5×5	1	tanh
Pool1	Avgppool	6	$14 \times 14 \times 6$	2×2	2	tanh
Conv-2	$1 \times$ Conv	16	$10 \times 10 \times 16$	5×5	1	tanh
Pool2	Avgppool	16	$5 \times 5 \times 16$	2×2	2	tanh
Flatten	Flatten	-	400	-	-	tanh
FC3	Dense	-	120	-	-	relu
FC4	Dense	-	84	-	-	relu
FC5	Dense	-	# of classes	-	-	softmax

Table 3. FC layers model.

Layer Name	Layer Type	Output Size	Activation
Input	Data	# of attributes	-
FC1	Dense	4000	relu
FC2	Dense	4000	relu
FC3	Dense	4000	relu
FC4	Dense	# of classes	sigmoid

The environment in which all development of our work had been processed is a $\times 64$ Ubuntu 20.04.4 LTS Operating System equipped with an Intel Core i7-11850H working at 2.5 GHz $\times 16$ and 32 GB DDR-4 RAM and a NVIDIA T1200 Laptop GPU (driver version: 510.47.03, CUDA version:11.6).

3. Proposed Approach

As cited above, the intention of this research was to realize the connection of relevances between different layers. To do so, we opted for applying the approach presented by [12] in two different FC layers. First, we applied it independently. To show that there is a direct relationship between neurons from different layers, we considered the relevance of the FRL and backpropagate it until the second FC layer that we pruned in the parallel process. In this way, we could see the effect of backpropagating the relevance throughout layers and see the correlation between them.

We applied the SLR approach proposed by [12] to obtain information about the most relevant parts forming the FRL. In this way, we were able to know the relevances for the final decision of each of the neurons comprising this last FC layer. To calculate the relevance propagated to the previous layers we used the insight introduced in Section 2.1.4 and multiplied each of the absolute value of weights that was connected with each neuron of the next layer with the relevance of these neurons from the next layer, for each neuron comprising the layer in question. Finally, after obtaining the relevances for each neuron from the layer, we sparsified the weight matrix of this layer the same way as for the FRL but while sparsifying the U matrix in the following way. We considered only the rows that obtained the highest value after the summation of multiplications of absolute weights of connections with each of the relevances of neurons connected from the next layer, instead of considering the original relevances of neurons as we implemented for the FRL.

At the same time, we carried out sparsification of the same number of layers only considering the independent relevances of each layer, following the criteria proposed by [12]. In this work, they present three different criteria to determine which elements of each layer were more relevant to the final decision of the network. Overall, the criteria based in the cost of weights was the most adequate to reduce the dimensionality of the problem and maintain the performance of the architecture to be as high as possible. The graphical representation of both approaches is given by Figure 1.

In case of VGG16, the FRL corresponds to FC7, and the backpropagation of the relevances has been carried out until FC6. FRL and previous FC layer of Lenet5 are FC4 and FC3, respectively. In case of the aforementioned three FC layers' architecture these layers are FC3 and FC2, respectively.

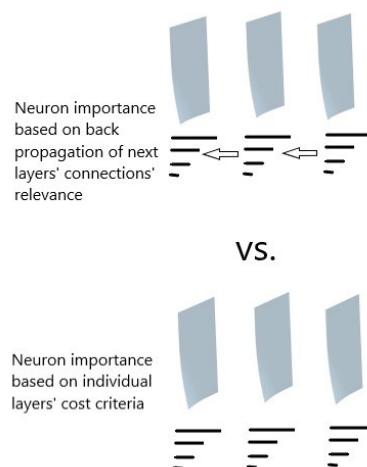


Figure 1. Comparison of the proposed approaches.

4. Experiment

In this section, details about the entire experimentation process are described. The results obtained are summarized as well.

Performance Metrics

Evaluation metrics used for determining which of the methods used is best for keeping the performance of the former network as high as possible are the accuracy vs. compression rate, AUC vs. compression, recall vs. compression, precision vs. compression, and specificity vs. compression, where the compression rate was defined in [12]. This last metric determines the relationship of the number of parameters between sparsified decomposed matrices and the original network's weight matrices. AUC is the area below the ROC curve—i.e., a graph showing the performance of a classification model at all classification thresholds. What is plotted in the curve is the FPR and TPR in the x and y axes, respectively, whose definitions are given in Equation (10) and (11). The definitions of the rest of the metrics mentioned above are given in Equations (7)–(9), where TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively. We used FRL's previous FC layer's compression rate to check the accuracy of the resultant network on different compression rate regimes.

Each of the variants proposed in this work, considering or not the relevance between layers, have been tested on well-known open source datasets for image recognition Cifar10, Cifar100, and MNIST. All of them have been trained using default 10,000 test images and 50,000 and 60,000 training images for the Cifar and MNIST datasets, respectively. To show their effectiveness in sensor related datasets, they were applied to the Room Occupancy Detection Dataset too. In this case, 1,000 samples were used for testing and the rest (9,129 samples) for training the network. In each case, we opted for establishing the same reduction rate (0.5) and sparsity rate (0.5) defined in [12], and we tested each variant with different rank k , which determines the number of columns and rows kept in the sparsified \hat{U} and \hat{V}^T matrices. We incremented the rank k until the performance metrics were equal to the ones obtained by the original network structure. In the testing phase 10 different seeds were established for testing each methodology in each dataset.

$$Accuracy(Acc) = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

$$Recall(Re) = \frac{TP}{TP + FN} \quad (8)$$

$$Precision(Pr) = \frac{TP}{TP + FP} \quad (9)$$

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN} \quad (10)$$

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \quad (11)$$

$$Specificity(Spec) = \frac{FP}{FP + TN} \quad (12)$$

5. Results

Figure 2 shows the accuracies obtained after testing both pruning techniques for the VGG16 architecture on the Cifar10 dataset. As is clear, there was no significant difference between the methods when applying an extremely low compression rate, which means that very few parameters of the original matrices were kept. Similarly, we could observe the same pattern when a higher number of parameters were kept in the original decomposed matrices, but there were significant differences between both compression rate regimes. In this case, applying the SLR method independently to different FC layers offers a higher

accuracy with the same compression rate, i.e., keeping the same number of connections between neurons.

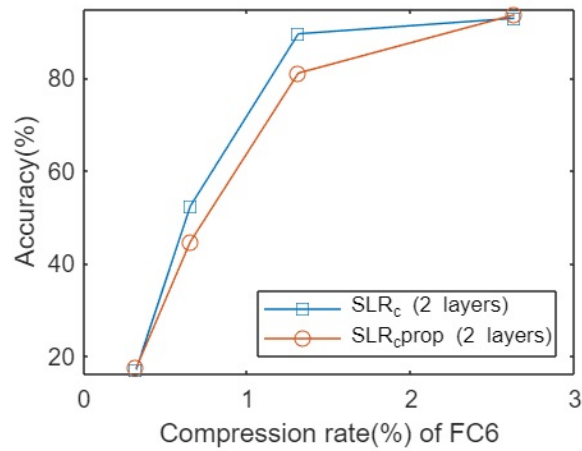


Figure 2. Accuracies for pruning VGG16 network on Cifar10 dataset for different compression rates.

Similarly, there could be the same pattern in case of Cifar100 dataset for the same network architecture. The SLR method was applied independently without any consideration of propagation of relevances across layers. In this case for lower compression rate regimes the difference is high as well. Figure 3 shows the results summarized for the Cifar100 dataset.

Simultaneously, Lenet5 architecture was pruned following both methodologies on MNIST dataset. Figure 4 shows the accuracies obtained for different compression rates. As could be observed, for the majority of the pruning rates applied when applying the SLR independently in different layers offers better performance than considering the backpropagation rule of the relevances from the FRL. However, in certain compression rate regimes, the last one outperforms the former one, but the difference is insignificant compared to the overall performance result.

Finally, both methodologies for pruning FC layers were adopted for pruning FRL and the previous FC layer of the Room Occupancy Detection dataset. Figure 5 shows the accuracies obtained for different compression rates. There is a clear tendency towards SLRProp in terms of accuracy, which determines that for the majority of the regions of compression SLRProp outperforms the SLR method.

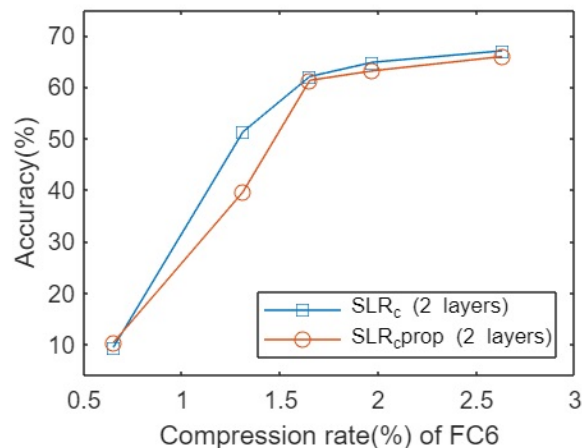


Figure 3. Accuracies for pruning VGG16 network on Cifar100 dataset for different compression rates.

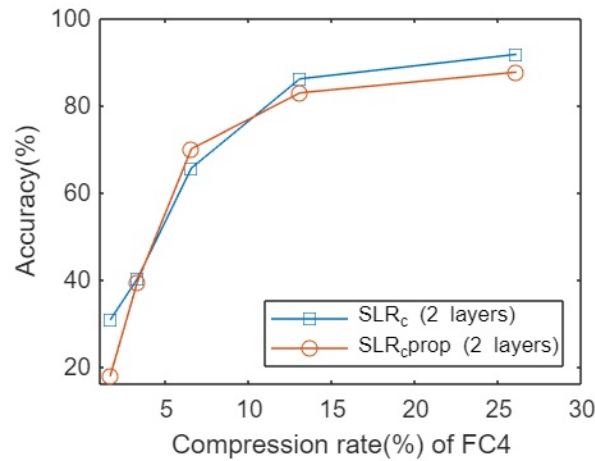


Figure 4. Accuracies for pruning Lenet5 network on MNIST dataset for different compression rates.

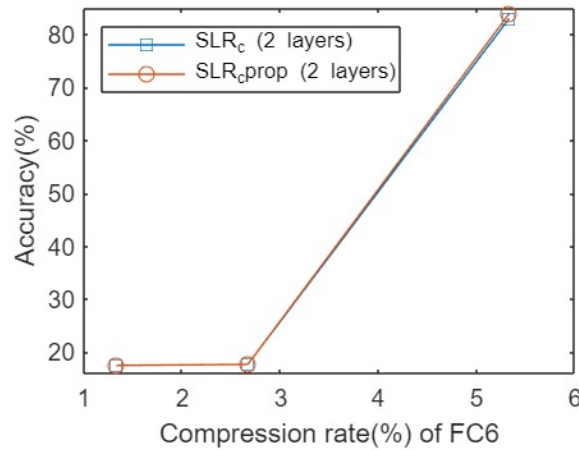


Figure 5. Accuracies for pruning FC layers architecture on Room Occupancy Detection dataset for different compression rates.

Overall, if we observe in detail each of the metrics defined in the previous section and given in Appendix A, we can obtain a general verdict about the performance of SLR and SLRProp applied to the four datasets described in Section 2. In 30 cases SLR obtained a higher metric value, and in 33 cases SLRProp obtained a better performance result. In 14 cases the results are identical for both methods. In Table 4 a brief review of these metrics is given. Once having this comparison, we can deduce that SLRProp's performance is slightly better than the original SLR that was presented by [12].

Table 4. SLR vs. SLRProp accuracies for different datasets.

Rank k	Cifar10-SLR	Cifar10-SLRProp	Cifar100-SLR	Cifar100-SLRProp	MNIST-SLR	MNIST-SLRProp	Room-SLR	Room-SLRProp
k = 2	17.037%	17.4074%	4%	2.8519%	21.084%	23.492%	17.5%	17.5
k = 4	52.3333%	43.5185%	9.8889%	8.8889%	38.006%	40.053%	17.7%	17.7%
k = 8	89.5926%	81.4444%	47.4444%	37.9259%	71.597%	80.469%	83%	83.9%
k = 16	92.963%	92.963%	63.4074%	62.4444%	93.258%	94.245%	-	-
k = 32	-	-	-	-	98.416%	98.385%	-	-

6. Discussion

As demonstrated in the previous section, the introduction of the concept of backpropagation of the relevances from the FRL to the rest of the layers of the original network does not always outperform the supposition of the relevances independently within layers. However, the general result shows that the SLRProp method is slightly better than the original version of sparsification presented by [12]. In this way, the breakthrough presented by [13] is preserved in this experiment; the final result the relevances propagated between different layers through the connections between neurons is of particular importance. For relatively high compression rate regimes, where the number of pruned connections is not very high, the performance metrics are almost identical for all architectures applied for the four different datasets. On the contrary, for very low compression rate regimes the performance metrics do not follow a distinguishable pattern, which shows the randomness of both methods when an excessive pruning is carried out in any of the mentioned architectures.

This shows that the components of each layer have certain influence on the rest of the network components, even though the main contribution to the final result of each component is more connected with other aspects than the connections' weights' absolute values across layers. In this case, the cost defined as the difference of the accuracy between the case when a certain component is eliminated from the original network and the original structure's accuracy showed that it could be more crucial when selecting which connections should be removed when pruning the original network. Additionally, backpropagating the relevances of the FRL to the previous FC layers could yield an even more adequate performance when applied to certain datasets, e.g., for the Room Occupancy Detection dataset. Consequently, this paper shows that the relevances propagated between layers play an important role when determining which are the most important components of the network structure.

To summarize, it is possible to say that the proposition presented by [13] echoed in Equation (6) is conserved in this experimental process, thus challenging its validity for every architecture of a convolutional neural network focused on image recognition. In fact, the ranking of the connection's relevance proposed by [12] offers an optimal result in terms of accuracy and network compression—needing only a very low percentage of parameters for representing sparsified matrices compared to the original network's matrices. However, the computational cost of calculating each matrices' components costs might be too high and ineffective in many scenarios, which creates the need for an alternative method for solving this issue of the training phase. The SLRProp alternative offers a slight improvement in different accuracy metrics, but is still too costly in terms of time efficiency and computational load. Attending these networks' weights' absolute values as a criteria to decide which columns and rows are maintained in the sparsified matrices offers a near identical result in terms of accuracy that needs $\sim 100\times$ less time for sparsifying the SVD matrices in the training phase. In applications where time response is crucial, this last alternative method may be more adequate.

Author Contributions: All the authors have contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data used in this work is available at <https://keras.io/api/datasets/> (accessed on 12 December 2022) and at <https://archive.ics.uci.edu/ml/index.php> (accessed on 16 December 2022) were a brief explanation of each dataset is given, as well as an explanation of how to use the data.

Conflicts of Interest: The authors declare no conflict of interest.

Sample Availability: Samples of the compounds are available from the authors.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area Under the ROC Curve
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
FC	Fully Connected
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
FRL	Final Response Layer
ML	Machine Learning
OBD	Optimal Brain Damage
OBS	Optimal Brain Surgeon
SLR	Sparse Low Rank Method
SVD	Singular Value Decomposition
TN	True Negatives
TP	True Positives
TPR	True Positive Rate

Appendix A

In this appendix the performance metrics described in this article for each of the datasets mentioned in Section 2 are given.

Table A1. SLR vs. SLRProp Cifar10.

Method-Rank k	Accuracy	AUC	Recall	Precision	Specificity
SLR-k = 2	17.037%	52.2763%	0%	0%	-
SLRProp-k = 2	17.4074%	50.6373%	0%	0%	-
SLR-k = 4	52.3333%	77.6934%	80.9155%	26.9259%	-
SLRProp-k = 4	43.5185	74.7603	62.4146%	17.4074%	-
SLR-k = 8	89.5926%	98.7243%	86%	92.0401%	-
SLRProp-k = 8	81.4444%	97.3784%	74.037%	87.3035%	-
SLR-k = 16	92.963%	99.1493%	92.9630%	93.2764%	-
SLRProp-k = 16	92.963%	99.1873%	92.9630%	93.5727%	-

Table A2. SLR vs. SLRProp Cifar100.

Method-Rank k	Accuracy	AUC	Recall	Precision	Specificity
SLR-k = 2	4%	68.0924%	0%	0%	100%
SLRProp-k = 2	2.8519%	65.6321%	0%	0%	100%
SLR-k = 4	9.8889%	80.7851%	0.0741%	0.6667%	99.9996%
SLRProp-k = 4	8.8889%	81.2581%	0.1481%	4%	100%
SLR-k = 8	47.4444%	96.5314%	18.3333%	86.0334%	99.9719%
SLRProp-k = 8	37.9259%	95.0432%	12.6667%	77.8508%	99.9637%
SLR-k = 16	63.4074%	96.2422%	54.7037%	76.9269%	99.8316%
SLRProp-k = 16	62.4444%	96.2004%	53.0741%	79.3686%	99.8578%

Table A3. SLR vs. SLRProp MNIST.

Method-Rank k	Accuracy	AUC	Recall	Precision	Specificity
SLR-k = 2	21.084%	58.7361%	20.914%	21.176%	91.3463%
SLRProp-k = 2	23.492%	59.857%	23.309%	23.5553%	91.5947%
SLR-k = 4	38.006%	74.2785%	37.41%	38.4775%	93.3527%
SLRProp-k = 4	40.053%	77.2657%	39.267%	40.6033%	93.6218%
SLR-k = 8	80.469%	94.7351%	80.219%	80.774%	97.8786%
SLRProp-k = 8	71.597%	92.8456%	70.98%	72.2031%	96.3698%
SLR-k = 16	93.258%	98.6997%	93.16%	93.3691%	99.2649%
SLRProp-k = 16	94.245%	98.856%	94.167%	94.3613%	99.3748%
SLR-k = 32	98.416%	99.7014%	98.393%	98.455%	99.8284%
SLRProp-k = 32	98.385%	99.7007%	98.366%	98.43%	99.8257%

Table A4. SLR vs. SLRProp Room Occupancy Estimation.

Method-Rank k	Accuracy	AUC	Recall	Precision	Specificity
SLR-k = 2	17.5%	49.8478%	17.5%	17.5%	72.5%
SLRProp-k = 2	17.5%	49.9085%	17.5%	17.5449%	72.5833%
SLR-k = 4	17.7%	62.6004%	17.5%	19.774%	76.3333%
SLRProp-k = 4	17.7%	63.3562%	17.5%	19.774%	76.3333%
SLR-k = 8	83%	93.9923%	49.8%	73.5598%	94.0333%
SLRProp-k = 8	83.89%	95.5216%	51.8%	77.54449%	95.08%

References

1. LeCun, Y.; Denker, J.; Solla, S. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*; Touretzky, D., Ed.; Morgan-Kaufmann: Burlington, MA, USA, 1989; Volume 2.
2. Hassibi, B.; Stork, D.; Wolff, G. Optimal Brain Surgeon and general network pruning. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; Volume 1, pp. 293–299. [\[CrossRef\]](#)
3. Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; Kautz, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. *arXiv* **2016**, arXiv:1611.06440. <https://doi.org/10.48550/ARXIV.1611.06440>.
4. Yu, C.; Wang, J.; Chen, Y.; Wu, Z. Transfer Channel Pruning for Compressing Deep Domain Adaptation Models. *Int. J. Mach. Learn. Cibern.* **2019**, *10*, 3129–3144. [\[CrossRef\]](#)
5. Muthukrishnan, R.; Rohini, R. LASSO: A feature selection technique in predictive modeling for machine learning. In Proceedings of the 2016 IEEE International Conference on Advances in Computer Applications (ICACA), Coimbatore, India, 24 October 2016; pp. 18–20. [\[CrossRef\]](#)
6. Yuan, M.; Lin, Y. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2006**, *68*, 49–67. [\[CrossRef\]](#)
7. Girshick, R. Fast R-CNN. *arXiv* **2015**, arXiv:1504.08083.
8. Sainath, T.N.; Kingsbury, B.; Sindhvani, V.; Arisoy, E.; Ramabhadran, B. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 6655–6659. [\[CrossRef\]](#)
9. Xue, J.; Li, J.; Gong, Y. Restructuring of deep neural network acoustic models with singular value decomposition. In Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH 2016), San Francisco, CA, USA, 8–12 September 2016; pp. 2365–2369.
10. Kholiavchenko, M. Iterative Low-Rank Approximation for CNN Compression. *arXiv* **2018**, arXiv:1803.08995.
11. On Compressing Deep Models by Low Rank and Sparse Decomposition. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 67–76. [\[CrossRef\]](#)
12. Swaminathan, S.; Garg, D.; Kannan, R.; Andres, F. Sparse Low Rank Factorization for Deep Neural Network Compression. *Neurocomputing* **2020**, *398*, 185–196. [\[CrossRef\]](#)
13. Yu, R.; Li, A.; Chen, C.F.; Lai, J.H.; Morariu, V.I.; Han, X.; Gao, M.; Lin, C.Y.; Davis, L.S. NISP: Pruning Networks using Neuron Importance Score Propagation. *arXiv* **2017**, arXiv:1711.05908.
14. Yeom, S.K.; Seegerer, P.; Lopuschkin, S.; Binder, A.; Wiedemann, S.; Müller, K.R.; Samek, W. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognit.* **2021**, *115*, 107899. [\[CrossRef\]](#)
15. Ayinde, B.O.; Inanc, T.; Zurada, J.M. Redundant feature pruning for accelerated inference in deep neural networks. *Neural Netw.* **2019**, *118*, 148–158. [\[CrossRef\]](#)
16. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both Weights and Connections for Efficient Neural Networks. *arXiv* **2015**, arXiv:1506.02626.
17. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
18. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

A.2 Task Offloading in Edge Computing using GNNs and DQN



ARTICLE

Task Offloading in Edge Computing Using GNNs and DQN

Asier Garmendia-Orbegozo¹, Jose David Nunez-Gonzalez^{1,*} and Miguel Angel Anton²

¹Department of Applied Mathematics, University of the Basque Country UPV/EHU, Eibar, 20600, Spain

²TECNALIA, Basque Research and Technology Alliance (BRTA), San Sebastian, 20009, Spain

*Corresponding Author: Jose David Nunez-Gonzalez. Email: josedavid.nunez@ehu.eus

Received: 11 September 2023 Accepted: 04 December 2023

ABSTRACT

In a network environment composed of different types of computing centers that can be divided into different layers (cloud, edge layer, and others), the interconnection between them offers the possibility of peer-to-peer task offloading. For many resource-constrained devices, the computation of many types of tasks is not feasible because they cannot support such computations as they do not have enough available memory and processing capacity. In this scenario, it is worth considering transferring these tasks to resource-rich platforms, such as Edge Data Centers or remote cloud servers. For different reasons, it is more exciting and appropriate to download various tasks to specific download destinations depending on the properties and state of the environment and the nature of the functions. At the same time, establishing an optimal offloading policy, which ensures that all tasks are executed within the required latency and avoids excessive workload on specific computing centers is not easy. This study presents two alternatives to solve the offloading decision paradigm by introducing two well-known algorithms, Graph Neural Networks (GNN) and Deep Q-Network (DQN). It applies the alternatives on a well-known Edge Computing simulator called PureEdgeSim and compares them with the two default methods, Trade-Off and Round Robin. Experiments showed that variants offer a slight improvement in task success rate and workload distribution. In terms of energy efficiency, they provided similar results. Finally, the success rates of different computing centers are tested, and the lack of capacity of remote cloud servers to respond to applications in real-time is demonstrated. These novel ways of finding a download strategy in a local networking environment are unique as they emulate the state and structure of the environment innovatively, considering the quality of its connections and constant updates. The download score defined in this research is a crucial feature for determining the quality of a download path in the GNN training process and has not previously been proposed. Simultaneously, the suitability of Reinforcement Learning (RL) techniques is demonstrated due to the dynamism of the network environment, considering all the key factors that affect the decision to offload a given task, including the actual state of all devices.

KEYWORDS

Edge computing; edge offloading; fog computing; task offloading

1 Introduction

Various computing centers can be found in a local networking environment, with possible interconnections. In the Edge Computing paradigm, this interconnection facilitates the transmission



of information is of particular interest. In many cases, when resource-constrained devices are allocated to solve computationally expensive tasks, they can become overloaded and not powerful enough regarding processability and memory availability. In this way, weaker computers can alleviate their computational load by assigning different tasks to more powerful devices nearby. These devices can vary depending on their complexity and proximity to these end-user devices. This variation of possible destinations is appropriate to distinguish different layers in an architecture, dividing it into the cloud, fog/edge, and IoT (Internet of Things) layers. The cloud tier comprises remote network servers rich in general resources with the capacity to store, manage, and process data. This general computer is the richest regarding processability and resource availability and often acts as a network orchestrator. In contrast, at the lowest level, it can find sensors, gadgets, and other IoT devices equipped with restricted computing capabilities but offer immediate responses to users. Its function is to collect information from the environment and act on environmental changes, among others. Meanwhile, other layers can be defined, such as the Fog and Edge layers, with greater capacities than the previous ones but with less processing and memory capacity than the cloud, being a valuable alternative for different types of computations.

Management and decision-making tools based on Artificial Intelligence (AI) algorithms have great potential to offer new and more efficient services that improve people's living conditions. These services could be of various types, from the classification of multiple classes of land cover [1] to systems where pollution forecasts are made [2]. These tools are possible thanks to the collection of information from the physical environment in real-time (RT) and the subsequent use of this data in complex Machine Learning (ML) and Deep Learning (DL) models. The models require high computability for the training phase and a large amount of available memory to store their parameters for the last inference. Consequently, IoT devices cannot store such an amount of data or train deep models, so they need to adjust the data and model sizes or send these mappings to more powerful devices. This could alleviate the problem of lack of resources faced by IoT devices, but a drop in accuracy would be inevitable. When end-user devices intend to perform certain calculations but are not equipped with sufficient resources or are overloaded, they have the opportunity to transmit their assigned tasks to other devices over the network given the interconnectivity between different nodes. A proper offloading strategy is crucial to avoid situations where certain nodes in an architecture absorb all tasks from nearby end devices. As a result, load balancing between nodes must be ensured and all tasks must be executed successfully.

In addition, depending on the environment in which this paradigm is located or in which the application is intended to be used, some alternatives will be more beneficial. For example, if there are latency requirements for tasks, streaming to the nearest nodes will be more appropriate than streaming to the cloud, although cloud servers are unlimited in memory and offer the highest processing capabilities. The weakness of using this alternative is that transmitting information from end-user devices to the cloud involves some delay and possible loss of information over the network. This loss could result from connection loss or other erroneous message information. The information can be vulnerable to intrusion attacks and inaccurate or incomplete. Transferring confidential information to the cloud is not the right decision because the vulnerability grows with increased information exposure over the network.

In contrast, IoT devices do not expose information over the network when they perform a task, making them the most secure option. The latency required by many applications also prevents using the cloud as a final computing center due to increased delay. The essential requirement of RT computing is immediate response, which is impossible to achieve using cloud computing. On the other hand, end-user devices offer immediate feedback, but their limitations can lead to a lack of model accuracy.

Excessively reducing the size of the models and the data needed to represent them leads to a severe drop in the performance of the resulting models. For example, a simple actuator has to give a specific response depending on the values in the environment. In order to obtain the answer, it can be necessary to apply an ML model that would not be feasible to compute on the IoT device or not with its original structure. However, the interconnectivity between different computers at different layers offers the possibility of computing these models in other computing centers, alleviating the computational load of these tiny devices.

In order to solve the problems mentioned above, it is essential to establish an appropriate task offloading policy. This would indicate in each case if it is necessary to carry out the download process, and if so, what the most appropriate destination will be in each case.

There are many alternatives intended to help with the task offloading decision problem. Some researchers opted for optimization algorithms. Other studies have chosen methods based on AI. Other alternatives, such as population-based and control theory methods, are outside the research interest.

This study applies Graph Neural Networks (GNNs) and Deep Q-Networks (DQN) to decide if it is feasible to offload a task from an end-user device to a richer computing center in terms of processability and available memory in each situation and to determine which is the best destination available in the area. It represents the local network structure in which different types of devices can be found in the Graph structure where each node is a computer, and the edges are the interconnections between them. This architecture can be extrapolated to an area of local networks where different IoT devices are interconnected with each other and with more powerful Edge Datacenters that offer the possibility of offloading tasks from small devices to Datacenters such as a Smart Home with small gadgets and a central router. In addition, the agent learns the network environment in the DQN learning process, where each action will be a decision to offload the task to one of the potential destination computers surrounding the source computer originating the task. Each state will represent each situation where all the characteristics of computers will be reflected. In each scenario, this research establishes a general remote cloud server that will serve as the orchestrator of the offload strategy and a fixed number of Edge Computing data centers and end-user devices or Edge devices. The proposed methods are evaluated by observing the success rate of the generated tasks, workload balance, and energy consumption. Finally, This study analyzes which computing nodes are most suitable for downloading the success rates of each device type.

The main contributions made in this work are the following. We offer a novel alternative to establish a task offloading strategy in a local network environment. The network architecture is almost replicated in the GNN architecture and the quality of a network connection for download issues has been rated with a novel parameter called download rating. Furthermore, environment updates are fully considered in the DQN learning process. Our methodology offers an innovative way to offload tasks in a local network environment, ensuring load balancing and completion of tasks within the desired latency. An overview of the procedure is given in [Fig. 1](#).

The rest of the paper is organized as follows. [Section 2](#) reviews some of the most representative works published in the literature. [Section 3](#) specifies the new algorithms proposed by this work. [Section 4](#) presents the materials and methodology applied in this work. In [Section 5](#), we carry out different experiments of the task offloading paradigm using a known simulator and the results are presented. In [Section 6](#), these results are analyzed and conclusions are reached.

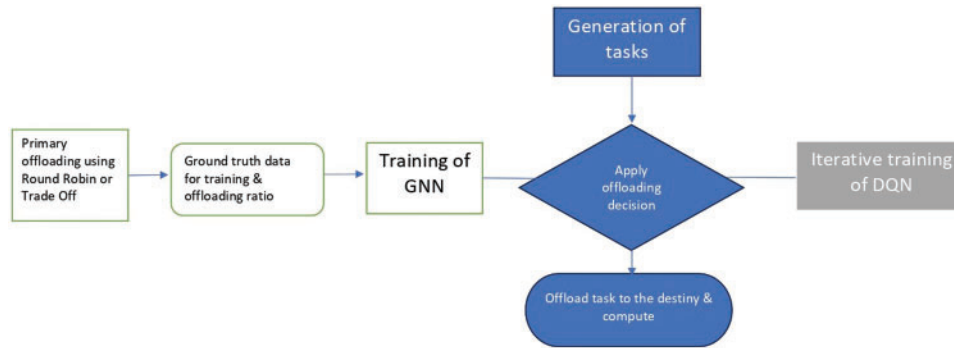


Figure 1: Overview of the entire process

2 State of the Art

Over the last decade, several researchers have found the task offloading paradigm a conflict of interest. The opportunity to transfer tasks from resource-constrained devices to resource-rich computing data centers can alleviate the computational load on end-user devices and complete tasks that were not feasible to complete at the source due to processing and memory constraints.

Those techniques have been widely used in different areas. Different techniques have been used in virtual reality (VR) applications, such as fog computing-based radio access networks (F-RAN) [3] or ML-based intelligent programming solutions [4]. In autonomous vehicle applications, task offloading techniques have been used to improve performance by reducing latency and transmission cost, as was done in [5]. Real-time traffic management was feasible by distributing decision-making tasks to Edge devices [6]. In the area of robotic task offloading, new paradigms have emerged, in [7], they presented an approach to simultaneous localization and mapping (SLAM) for RGB-D cameras like the Microsoft Kinect, and in [8], a novel Robot-Inference-and-Learning-as-a-Service (RILaaS) platform for low-latency and secure inference serving of deep models that could be deployed on robots was introduced. Nonetheless, there are already commercial solutions for offloading tasks in robotics [9–11]. Similarly, cloud-based solutions can be found in video streaming applications [12, 13]. However, offloading Edge should improve performance as in [14, 15], by enabling gateways and facilitating caching and transcoding mechanisms, respectively. The challenge of transferring computationally expensive tasks to Edge nodes has been addressed in [16–18] in the area of disaster management, but is still underexplored in this field. In the IoT field, task offloading has been of special interest since its inception, since these devices with limited resources often face this drawback. Due to the long delays involved in network transfer between IoT devices and the cloud, edge offloading needs to be considered. The collaboration between IoT devices and Edge devices could be useful in the area of smart health, being a good alternative to help paralyzed patients [19]. However, due to the growing number of IoT devices, the best option would be the collaboration between the Edge and Cloud servers as they did in [20] proposing a paradigm that foresees an IoT Cloud Provider (ICP)-oriented cooperation, which allows all devices that belong to the same public/private owner to participate in the federation process.

Different strategies have been proposed to solve the problem of task offloading. Optimization algorithms have become a very useful and frequently used solution for this paradigm. Mixed integer programming (MIP) has become a useful tool for resource allocation problems, addressing network synthesis and allocation issues [21]. In other words, they opted for greedy heuristic solutions [22–24] to

solve the task offloading problem. The main advantage of these is that they offer a low execution time, they do not require specialized optimization tools for their resolution and rather they can be expressed as pseudocode, easily implementable in any programming language. These become much more efficient when the task offloading problem is modeled as a nonlinear constrained optimization problem, or when the scale of the scenario is large enough [25]. In this case, a greedy heuristic could estimate the exact solution [13,22,24]. In other words, game theory was chosen, formulating the problem of partial task offloading in a multi-user infrastructure, Edge Computing and multi-channel wireless interference environment as an offloading game [26]. The Cloud-Edge game could be seen as an infrastructure game in which the players are the corresponding infrastructures [27]. Contract theory [28–30] and local search [31,32] are another type of optimization solutions for the task offloading problem.

Another interesting approach to solving the problem of task offloading is the use of methods based on AI. This branch includes all ML methods, including Supervised Learning, Unsupervised Learning, DL, and Deep Reinforcement Learning (DRL) methods. The download destination could be chosen following the simplest models, such as a regression model [33] or regression trees [34]. However, given the dynamism of network environments, modeling has been performed with the support vector regressor [35] and the nearest neighbor regressor [36] for future load prediction and energy efficient utilization of the Edge servers, respectively. In [37], a resource-aware offloading video analysis in Mobile Edge Computing and a resource-aware offloading (ROA) algorithm using the radial basis function networks (RBFN) method to improve reward were proposed under the resource deadline constraint. Taking into account unsupervised models, clustering models are useful tools to group resources depending on the distance between computing nodes [38] and task demands [39] and analyze the allocated resources [40].

DL can be an accurate tool for making task offloading decisions, based on the resource usage of the processing Edge nodes, the workload, and the quality of services (QoS) constraints defined in the Service Level Agreement (SLA) [41]. In [42], a new multi-objective strategy based on biogeography-based optimization (BBO) algorithm for Mobile Edge Computing (MEC) offloading was proposed to satisfy multiple user requirements (execution time, power consumption, energy, and cost). In [43], a task offloading model based on dynamic priority adjustment was proposed. Second, a multi-objective optimization model for task scheduling was constructed based on the task offloading model, which optimizes the time delay and energy consumption. In [44], they proposed an Improved Gorilla Troops Algorithm (IGTA) to offload dependent tasks in MEC environments with three objectives: minimizing the application execution latency, the power consumption of light devices, and the used cost of MEC resources. DL models have been used to minimize the computational load under dynamic network conditions and constrained computational resources [45]. A model that also considers the challenges of speed, power, and security, while satisfying QoS with dynamic needs, has been proposed to determine the combination of different computing nodes [46]. In [47], they developed a novel calibrated contextual bandit learning (CCBL) algorithm, where users learn the computational delay functions of micro base stations and predict the task offloading decisions of other users in a decentralized manner. At [48], they presented a novel federated learning framework for GAN, namely Collaborated g Ame Parallel Learning (CAP), which supports parallel training of data and models for GAN and achieves collaborative learning between edge servers, devices, and Cloud. Furthermore, they proposed a Mix-Generator (Mix-G) module that splits a generator into the sharing layer and the personalizing layer.

DRL techniques have emerged as an interesting alternative to typical task-offloading policies. Deep Q networks have been used to solve the task offloading problem [49] and have been optimized by introducing a short-term memory (LSTM) [50] into them. An intelligent partial offloading scheme was

proposed in [51], namely digital twin-assisted intelligent partial offloading (IGNITE), which combines the improved clustering algorithm with the digital twin (DT) technique, in which unreasonable decisions can be avoided by reducing the size of the decision space and finding the optimal offloading space in advance. In the same field, reference [52] proposed a mobility-dependent task offloading (MESON) scheme for urban vehicle edge calculation (VEC) and developed a DRL-based algorithm to train the offloading strategy. To improve the training efficiency, a vehicle mobility detection algorithm was further designed to detect the communication time between vehicles and Road Side Units (RSUs). In this way, MESON was able to avoid unreasonable decisions by reducing the size of the action space. Finally, the DRL algorithm was used to train the offloading strategy. In [53], they used the Markov decision process (MDP) that minimizes the total completion time. In [54], they considered a wireless MEC system that governs a binary offloading decision to execute the task locally on the Edge devices or on the remote server, proposing a Reinforcement Learning-based Intelligent Offloading online (RLIO) framework that adopts the optimal offloading policy.

Other approaches that differ from those mentioned above include population-based methods and control theory-based methods. Swarm Intelligence methods [55,56] and Evolutionary Algorithms [57,58] are the two variants of population-based methods that have been proposed to address the problem. Solutions based on control theory include optimal control [59,60], state feedback control [61] and Lyapunov optimization processes [62] among others.

Most of the mentioned studies implemented using an outdated methodology that has been surpassed by recent models such as deep models or Reinforcement Learning (RL) models, or those that chose to use these techniques are single objective and/or do not care about task features and the actual workload of the destinations. In contrast, this study applies a simplistic approach that considers the nature of the tasks and the updated status of potential download destinations, facilitating user understanding while achieving high accuracy and competitive performance. It provides a methodology representing the network architecture in a graph and an RL technique that considers all the key factors when determining the optimal download decision. The research proposes a novel feature to evaluate the goodness of a download destination, which is a critical factor in determining whether a potential download route is valuable for a given task. Table 1 compares the latest and most relevant works, specifying the methodology proposed in each work.

Table 1: Comparative of current works on edge computing

Work	Proposed method	Field
[24]	Heuristic greedy offloading scheme	Multi-access mobile edge computing
[28]	Contract theory. Negotiation between task publisher and fog nodes as an optimization problem	Fog computing
[33]	Multi-task regression problem & Multi-task learning based feedforward neural network (MTFNN) model	Multi-access edge computing
[34]	Module placement method by classification and regression tree algorithm (MPCA) & Probability of network's resource utilization in the module offloading (MPMCP)	Mobile fog computing
[37]	Radial basis function networks-based resource-aware offloading	Video analytics in mobile edge computing

(Continued)

Table 1 (continued)

Work	Proposed method	Field
[38]	Balanced clustering and joint resources allocation (BCJRA)	Mobile fog computing
[39]	Dynamic mobile cloudlet cluster policy (DMCCP)	Fog computing
[40]	Server partitioning algorithm based on clustering. Multi-user game with Nash equilibrium	Mobile edge computing
[41]	Neural networks for mapping quality of service required levels and (expected) application workload to concrete resource demand	Edge computing
[42]	Multi-objective strategy based on the biogeography-based optimization (BBO) algorithm	Mobile edge computing
[43]	Task unloading model based on dynamic priority adjustment & Multi-objective optimization model	Task offloading & Real-time scheduling
[44]	Improved Gorilla troops algorithm (IGTA)	Multi-access edge computing
[46]	Deep learning-based dynamic task offloading in mobile cloudlet (DLDTO)	Mobile computing
[47]	Novel calibrated contextual bandit learning (CCBL) algorithm	Mobile edge computing into an ultra-dense network (UDN)
[48]	Collaborated game parallel learning (CAP) for GANs & Mix-generator module (Mix-G) that divides a generator into the sharing layer and personalizing layer	
[49]	Deep Q-learning approach for designing optimal offloading schemes, jointly considering selection of target server and determination of data transmission mode	Mobile edge computing
[50]	DRL & LSTM network layer and the candidate network set	Mobile edge computing
[51]	Digital twin-assisted intelligent partial offloading (IGNITE)	Vehicle edge computing
[52]	Mobility-aware dependent task offloading (MESON) scheme for urban VEC and a DRL-based algorithm to train the offloading strategy	Vehicle edge computing
[53]	Markov decision process	IoT & Edge computing
[54]	Reinforcement learning based intelligent online offloading (RLIO)	Mobile edge computing
[55]	Fuzzy clustering	Mobile edge computing

3 Proposed Algorithms

This study proposes a well-known DRL technique using GNN and DQN to solve the task offloading problem. Brief descriptions of both architectures are given in this section. Finally, the

training processes of both algorithms are explained utilizing Fig. 2, showing the complete architecture procedure.



Figure 2: Training procedure of the entire architecture divided in different steps

3.1 Graph Neural Network

Graphs are a data structure representing a collection of elements (nodes) and their connections (edges). A GNN is a type of neural network (NN) that works directly with the graph's structure. In the used case, each node in the network represents a computing center that can be an IoT/Edge device, an Edge server, or a cloud server. Graphs are a data structure representing a collection of elements (nodes) and their connections (edges). The edges between these nodes represent the connections between the different computing centers, which can be the download paths of the tasks that must be completed to meet their requirements.

Each node represents each computing device, a potential destination for the download task in question. Furthermore, the edges represent the connection between these devices, whose characteristics are as follows. The characteristics of each node are determined by the computing device's available RAM, millions of instructions per second (MIPS), central processing unit and memory, and the desired task latency and file size in bits associated with the task. The characteristics of Edge are determined by the offload classification defined in this work, that is, the number of tasks successfully executed using the offload path divided by the total number of tasks offloaded using the path.

The output size of the network will be determined by the number of possible destinations of the task initially assigned to the IoT/Edge device. The number of output neurons will be equal to those possible destinations. The output would be binary, downloading/not downloading to each possible destination.

To train the network, we apply the real data produced by the architecture following two well-known offloading algorithms, Trade-Off and Round Robin. The download destinations for each task obtained following any of the mentioned algorithms would be the actual data used to train the network. Once the network is trained, the input would be the task with its characteristics and the output would be a binary decision of the possible download destinations. A brief description of our algorithm is provided in Algorithm 1.

Algorithm 1: GNN

```

Nodes ← Available computing centers
Edges ← Connection between computing centers
NodeFeatures ← RAM, Mips, CPU, latency, file size
for NTasksExecuted Satisfactory do
    DestinyNode ← TradeOff / Round – Robin(task)
    EdgesSuccessfullyExeceededTasks ← EdgesSuccessfullyExeceededTasks + 1
end for
for NEdges do
    EdgeFeature ← EdgesSuccessfullyExeceededTasks / NTasksOffloadedbyEdge
end for
for Ntasks do
    Output ← GNN(task)
    Loss ← CrossEntropyLoss(Output, DestinyNode)
end for
for Ntasks do
    OffloadingDestiny ← GNN(task)
end for

```

3.2 Deep Q-Network

RL is a framework in which the agent attempts to learn from its environment by obtaining different rewards on each action performed in that environment. The agent's objective is to maximize the sum of rewards obtained by performing consecutive actions following its policy, and by optimizing this policy the problem in question is solved. After obtaining an observation of its environment (s_t) the agent acts a_t following its policy $\pi(a_t|s_t)$. Consequently, depending on the action performed in that observation, a reward and the next observation (s_{t+1}) are obtained.

DQN was developed by [63]. Deep neural networks (DNN) and replay techniques were used to optimize the Q-learning process. Q-learning is based on the function Q that measures the expected return or the discounted sum of rewards obtained from state s by taking action a first and following policy π . An optimal function Q^* is defined and, using the Bellman optimization equation (see Eq. (1)) as an iterative update, convergence of the function Q is guaranteed.

$$Q_{i+1}(s, a) = E[r + \gamma * \max_{a'} Q_i(s', a')] \quad (1)$$

Representing the function Q by combining all possible actions and states is not the most practical option in most cases. For this reason, a function approximator is used for this. Using the NN approximation can be done using parameters θ and minimizing the loss function.

$$L_i(\theta) = E_{s,a,r,s'}[(y_i - Q(s, a; \theta_i))^2] \quad y_i = r + \gamma * \max_{a'} Q(s', a'; \theta_{i-1}) \quad (2)$$

In the use case, the actions were the possible decision to download to each of the potential destinations on the network, given the state of the environment. The state of the environment will be determined by the task's characteristics and each device's state and capabilities. The properties of the task that conditioned the state of the environment were the maximum allowed latency and the size of the file in bits belonging to the task. Similarly, each computing device's available RAM, MIPS, central processing unit, and memory determined the rest of the state properties. If the task requirements were successfully met, the reward for downloading to a given computing center would

be 1, and -1 if the requirements for that action were not met. Following the technique above, the optimal download policy was obtained. Finally, the optimized policy would determine the optimal download destination for each task. Algorithm 2 summarizes the method.

Algorithm 2: DQN

```

for N tasks do
   $s \leftarrow$  RAM, Mips, CPU, memory, latency and file size
  for N possible destinies do
     $a \leftarrow$  Possible destiny
    Calculate  $L(\theta_i)$ 
  end for
   $a \leftarrow \max_a Q(s, a; \theta)$ 
   $Q_{i+1} \leftarrow Q(s, a)$ 
end for

```

3.3 Training Procedure & Orchestration of Tasks

In the case of GNN, it is first necessary to perform a training process following any of the two default methods available in the simulator. In each iteration, any devices that make up the IoT layer will randomly create a task. All devices will send a message to the cloud reporting their actual status, even if they have a task to solve (Step 1 in Fig. 2). In this scenario, the cloud will orchestrate the download action following the predetermined algorithm by sending a message to the device (Step 2 in Fig. 2), and this will be downloaded to the destination (Step 3 in Fig. 2). After all, if the task has been completed by meeting the requirements, that will be a positive result for the subsequent training of the GNN; otherwise, it will be negative. Once the entire training procedure of the default algorithm is completed, the GNN will use the download decisions and the output generated in the previous step as ground truth and perform the training process after defining the download rating for each network connection. As an edge feature. For both training procedures, the graph shape was determined by the network structure (influenced by the number of IoT devices), the learning rate was 0.001, the optimizer was Stochastic Gradient Descent (SGD), and 10000 was the number of epochs. The GNN training will be conducted through the cloud. Finally, the cloud will decide to download after each device sends the message with the information about its status (Step 1 in Fig. 2), and the cloud will return the message to the task-generating device informing about the download destination (Step 2 in Fig. 2). After sending the task to the target device (Step 3 in Fig. 2) and completing the task on this device, the results will be sent back to the source device (Step 4 in Fig. 2).

In the case of DQN, each device will send information about its status to the cloud (Step 1 in Fig. 2). There, taking the state of the environment based on the offloading policy, the optimal action must be taken. If the task was completed meeting the requirements, the reward will be 1, and 0 otherwise. In this way, an optimal offloading policy will be obtained after converging the Q function.

Finally, the Q function obtained will determine the optimal download destination in the cloud after each device sends its state to the cloud, and it returns the message to the task generator indicating where to download its assigned task (Step 2 in Fig. 2). After sending the task to the target device (Step 3 in Fig. 2) and completing the task on this device, the results will be sent back to the source device (Step 4 in Fig. 2).

4 Material & Methodology

This section explains the environment in which the methodologies presented in the previous section were applied in the experimental process. The software and hardware used in the experiments are also described.

For the experimental processes we chose to use a well-known Edge Computing simulator called PureEdgeSim [64]. The simulator offers high configurability through its modular design. In this way, by editing each module and adjusting it to the user's needs, it is simple and feasible to reproduce the desired environment in each case.

The hardware environment in which all development of our work took place is a $\times 64$ Ubuntu 20.04.4 LTS Operating System equipped with an Intel Core i7-11850H working at 2.5 GHz $\times 16$ and 32 GB DDR-4 RAM and a NVIDIA T1200 Laptop GPU (driver version: 510.47.03, CUDA version:11.6).

The study established between 10 and 30 end-user devices in this case, forming the IoT-Edge layer. It repeated the experiment 3 times and compared the results of applying the abovementioned algorithms to make task-offloading decisions. These devices were dynamic, and their range of motion was limited to 200×200 units. The Fog-Edge layer comprised four data centers, each located symmetrically in the coverage area. Each of these Edge Data Centers covered an area of 100×100 units. Finally, a resource-rich cloud platform offered greater computing and memory.

Each of the end-user devices was interconnected with each other. In this way, interconnections between them were feasible. Similarly, each of these end-user devices was connected to the nearest Edge Datacenters, and all were connected to the cloud.

The orchestrator of the decision to download was the cloud. It was equipped with 200 cores, 40,000 MIPS, 16 GB of RAM, and 1 TB of memory.

The Edge Datacenters were equipped with ten cores, 40,000 MIPS, 16 GB of RAM, and 200 GB of memory. Its idle power consumption was 100 Wh, with a maximum consumption of 250 Wh.

Finally, the number of Edge devices or end-user devices was 10, 20, and 30 in each experimental test. Their operating system was Linux and they had an architecture of $\times 86$. These devices had dynamic behavior in some cases, with a speed of 1.8 m/s. The type of network connection used to interconnect with the rest of the devices was WiFi with a bandwidth of 1300 Mbits/s, with a latency of 0.005 s. There were 5 different types of Edge devices and their characteristics are summarized in Table 2.

Table 2: Characteristics of different types of Edge devices

Device type	Type 1	Type 2	Type 3	Type 4	Type 5
Speed (m/s)	1.8	0	0	0	1.8
Pause duration (m/s)	100–400	0	0	0	100–400
Mobility duration (m/s)	60–100	0	0	0	60–100
Battery powered	Yes	No	Yes	No	No
Battery capacity (Wh)	18.75	–	56.2	–	–
Initial battery (%)	100	–	100	–	–
Idle energy consumption (Wh)	0.2	3.8	1.7	0.4011	0.4011
Max. energy consumption (Wh)	5	5.5	23.6	0.436	0.436
Cores	8	4	8	0	0

(Continued)

Table 2 (continued)

Device type	Type 1	Type 2	Type 3	Type 4	Type 5
MIPS	25000	16000	110000	0	0
RAM (GB)	4	4	8	0	0
Storage (MB)	256	128	256	0	0
Percentage	18	11	11	28	32

Each device could spawn any applications or tasks whose specifications are summarized in [Table 3](#). Container size refers to the size of the application in kB. The request size refers to the download request sent to the orchestrator and then to the device where the task will be downloaded in kB. Result size refers to the downloaded task results in kB.

Table 3: Characteristics of different types of tasks

Application type	Hard real-time	Soft real-time	Non real-time
Generation rate/s	20	30	3
Latency (s)	0.02	0.5	300
Task length (Millions of instructions)	500	5000	30000
Container size (kB)	20	1500	2200
Request size (kB)	20	1500	2500
Results size (kB)	20	50	200
Percentage	20	30	50

This study applied the offload decision algorithms against the default methods provided by the simulator, Round Robin, and Trade-Off. It introduced different options regarding possible download destinations by including all devices, Edge devices only, Edge Data Centers only, Edge Data Centers and cloud-only, Edge devices and cloud-only, and Edge devices and Edge Data Centers only.

In total, there were 6 offloading configurations \times 3 number of Edge device possibilities \times 4 algorithms = 72 simulation configurations. Each simulation time was established at 200 s.

5 Experiments & Results

In the experimental process, we considered the following parameters: energy consumption, tasks executed in each layer, and success rate. Additionally, we considered the distribution of the workload among different devices. Task failure could be due to different reasons, such as lack of available memory, violation of latency constraints, or network traffic congestion. His explanation is given below:

- Success rate: The ratio of the number of successfully executed tasks divided by the total number of tasks.
- Energy consumption: The power consumed by all devices of each type during each experimentation process.

- Workload distribution: Refers to the number of tasks distributed by each type of device in each experimentation process.

5.1 Tests with 10 Edge Devices

First, only ten devices were placed in the end-user layer, and these devices were divided into various types following the percentages shown in Table 2. These randomly generated the three types of tasks following the percentages and generation rates listed in Table 3. The success rate results are depicted in Table 4.

Table 4: Success rate of different algorithms including different types of destiny devices (10 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.9194%	100%	66.0484%	99.9194%	54.5161%	100%
Round Robin	99.8387%	100%	47.1774%	99.8387%	55.4839%	100%
GNN	99.9194%	100%	80.4032%	99.9194%	59.1935%	100%
DQN	99.9194%	100%	82.0968%	100%	65.5645%	100%

As can be seen, the most critical environments were when there were no Edge Data Centers available as possible download destinations. This could be because the Edge devices were not equipped with sufficient capabilities to computationally support the rest of the devices' tasks. Likewise, the cloud was too far from these end-user devices, so latency requirements were not met in most cases where the cloud was the download destination. Those problems were solved when tasks were offloaded to Edge Data Centers, which are computationally less powerful than the cloud platform but still have high capabilities. In the same way, being located close to these Edge devices, task latency was not an issue.

The energy consumption of Edge devices and Edge Data Centers are shown in Tables 5 and 6, respectively. Energy consumption was higher in cases where there were no Edge Data Centers available as possible download destinations, and the algorithm chosen to decide the download destination was GNN or DQN. In these cases, because the best download destinations were not available, the Edge devices that can perform the tasks consume more power. However, in the default algorithms, this is not the case. Most of the tasks could have been offloaded to the cloud, which hurts the success rate, as seen in Table 4. Edge Data Centers show a similar consumption pattern for all algorithms, slightly lower for all download policies when they were not potential download destinations.

Table 5: Energy consumption in Wh of Edge devices for different algorithms including different types of destiny devices (10 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	3.1947	3.2359	3.9221	3.1246	3.5854	3.1246
Round Robin	3.1032	3.1246	3.1184	3.1246	3.6223	3.1246
GNN	3.1745	3.2707	4.3367	3.1246	3.6223	3.1246
DQN	3.1487	3.1902	4.5019	3.1246	3.6406	3.1246

Table 6: Energy consumption in Wh of Edge Datacenters for different algorithms including different type of destiny devices (10 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	34.4092	34.7847	33.8889	34.5972	33.6111	34.9722
Round Robin	34.3193	34.9722	33.8889	34.5966	33.8889	34.9722
GNN	34.3577	34.6034	33.8889	35.0087	33.8889	34.9722
DQN	34.4162	34.8765	33.8889	35.0125	33.8889	34.9722

5.2 Tests with 20 Edge Devices

We repeated the experiment from the previous subsection by changing the number of Edge devices to 20. The results of the success rates are shown in [Table 7](#).

Table 7: Success rate of different algorithms including different type of destiny devices (20 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.8641%	100%	61.8478%	99.8913%	97.1739%	100%
Round Robin	99.8370%	99.7011%	96.30434%	99.8370%	59.9185%	100%
GNN	99.9185%	100%	84.4837%	99.9185%	97.1739%	100%
DQN	99.9457%	100%	84.3478%	99.9728%	97.1739%	100%

The success rate trend continued when we doubled the number of Edge devices. However, as the number of free Edge devices must have been higher than in the previous experiment the success rates were higher when Edge devices were involved and not Edge Data Centers. In cases where Edge Data Centers were possible destinations, rates are 100% or close to it. In this test, when Edge Datacenters were not potential destinations, performance degradation was observed when Edge devices were the only potential destinations for the Round Robin algorithm and when the cloud was also a potential offload destination for the Trade-Off algorithm. In these cases, the Edge devices would not be sufficient to respond to the tasks and the cloud would not respond within the desired latency, respectively.

[Tables 8](#) and [9](#) show the energy consumption of edge devices and Edge Data Centers, respectively.

Table 8: Energy consumption in Wh of Edge devices for different algorithms including different types of destiny devices (20 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	8.3796	7.7330	8.7102	7.0386	9.3050	7.0386
Round Robin	7.6535	6.9917	8.6272	7.0386	8.0294	7.0386
GNN	7.6113	7.2317	8.7088	7.0386	9.3050	7.0386
DQN	6.6980	7.5004	8.7049	7.0386	9.3286	7.0386

Table 9: Energy consumption in Wh of Edge Datacenters for different algorithms including different type of destiny devices (20 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	34.9217	35.4306	33.8889	36.1160	33.8889	36.6806
Round Robin	34.9298	35.9590	33.8889	35.9284	33.8889	36.6806
GNN	34.9240	36.6409	33.8889	36.1788	33.8889	36.6806
DQN	34.3268	35.7012	33.8889	36.2064	33.8889	36.6806

There is no evident variation in the power consumption of Edge Data Centers when doubling the number of Edge devices generating tasks. However, the power consumption of the Edge devices was almost double that in the previous case since more tasks were generated while the number of Edge Data Centers was fixed. Since computing platforms with higher capabilities were the same for a larger number of tasks, more tasks were offloaded to devices with limited resources. As in the previous experiment, Edge device consumption was slightly higher for cases where Edge Data Centers do not receive any tasks.

5.3 Tests with 30 Edge Devices

Finally, we replicated the experiment from the previous subsections by changing the number of Edge devices to 30. The results of the success rates are shown in [Table 10](#).

Table 10: Success rate of different algorithms including different types of destiny devices (30 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	99.9324%	99.8986%	70.2196%	99.8986%	89.0541%	100%
Round Robin	99.1892%	99.4595%	91.5541%	99.8311%	99.3986%	100%
GNN	99.9493%	100%	92.3142%	99.9324%	93.9696%	100%
DQN	99.9662%	100%	94.0372%	99.9662%	94.5777%	100%

Overall the success rates are better than with 10 Edge devices but comparable to the case of 20 devices.

The power consumption of the Edge devices and Edge Data Centers are shown in [Tables 11](#) and [12](#), respectively. The increase was proportional to previous cases, with a low variation in the consumption of Edge Data Centers and a significant variation in the consumption of Edge devices. Apart from the lineal increase in consumption due to the higher number of tasks computed in these types of devices, in the case when Edge devices and Cloud where potential destinies the highest energy consumptions were reported when DQN and GNN were the applied algorithms, as consequence of a higher number of tasks offloaded to Edge devices than to the cloud. As in the previous cases, when a greater proportion of tasks were offloaded to Edge devices (higher energy consumption) the success rates were higher, due to the latency violation that occurred when the cloud was in charge of performing the task.

Table 11: Energy consumption in Wh of Edge devices for different algorithms including different types of destiny devices (30 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	14.1875	11.6458	12.4942	10.6673	13.8637	10.6673
Round Robin	10.7567	12.1466	12.9142	10.6673	13.8015	10.6673
GNN	12.9003	12.0004	13.2908	10.6673	13.8015	10.6673
DQN	11.9993	11.8096	13.4561	10.6673	13.8015	10.6673

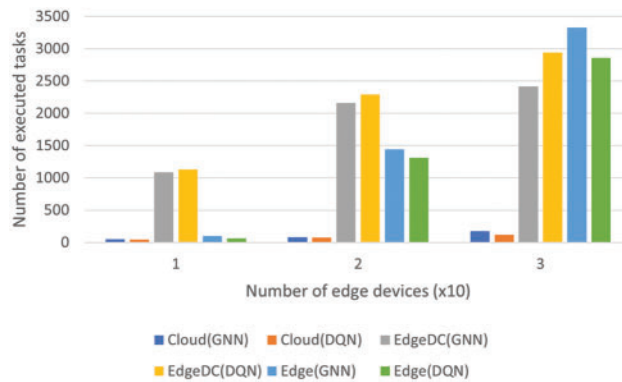
Table 12: Energy consumption in Wh of Edge Datacenters for different algorithms including different types of destiny devices (30 Edge devices)

Algorithm	All devices	Edge & Edge DC	Edge & Cloud	Edge DC & Cloud	Edge	Edge DC
Trade Off	34.8949	36.0532	33.8889	37.1575	33.8889	38.0972
Round Robin	35.6297	36.4035	33.8889	36.9680	33.8889	38.0972
GNN	35.1027	36.3284	33.8889	37.0880	33.8889	38.0972
DQN	35.9075	36.1982	33.8889	37.2137	33.8889	38.0972

5.4 Task Distribution with Varying Edge Devices

Next, this study established all types of computing devices as possible offloading destinations, and by varying the number of Edge devices between 10 and 30, as in the previous tests, this research observed the distribution of the download destinations in each case. The algorithms considered were GNN and DQN.

Fig. 3 indicates that the incremental trend toward computing tasks on Edge devices remains for both algorithms as the number of Edge devices grows and, consequently, the number of generated tasks does as well. This agrees with the increase in energy consumption observed in the previous sections. For both algorithms, the Edge Data Centers cannot attend to more tasks in the case of 30 Edge devices, relegating the rest of the tasks to the Edge devices, consequently having them attend to more tasks.

**Figure 3:** Task distribution with different algorithms and number of Edge devices

5.5 Success Rate of Different Layers

Finally, we established as possible offloading destinies all types of computation devices and varied the number of Edge devices between 10 and 30 as in previous tests we observed the success rate of different layers, to determine the optimal destination for computing the tasks generated by end-user devices. The algorithms regarded were GNN and DQN.

In this case, the worst results were given by the cloud platform. Although in terms of computational capabilities, it is the best option compared to the rest of the devices, the latency requirements were more difficult to meet due to the long time required to cross the entire network. This was expected. However, Edge devices were as good as Edge Datacenters in terms of accuracy for 10 and 20 devices. In the latter case, where 30 Edge devices were generating tasks, the Edge Data Centers were fully occupied, so more tasks were offloaded to resource-constrained devices. In isolated cases, the Edge devices were not able to complete the task, which is the reason why in the last case the Edge devices did not reach 100% accuracy for both algorithms. Fig. 4 shows the success rates for each layer & algorithm with different numbers of Edge devices.

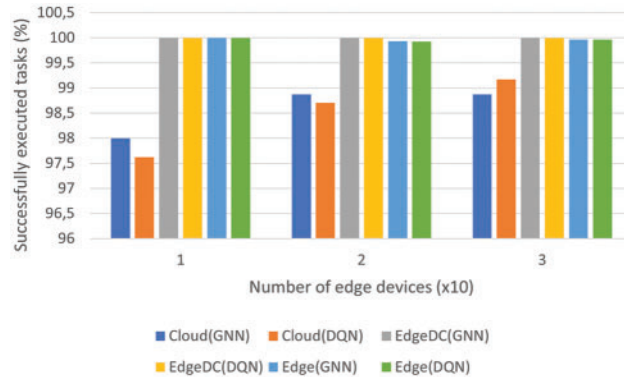


Figure 4: Success rates with different algorithms and number of Edge devices

6 Discussion & Conclusion

Section 5 presents the results obtained in the experimental process and observes different parameters.

Regarding success rate, there is a clear trend in favor of cases where Edge Data Centers were included in the download destinations. This is because they had sufficient computability and were located close to the Edge devices from where the tasks were generated. The worst results were obtained when only the cloud was available as a powerful computing center. In this situation, where network traffic congestion will have caused longer delays in task responses, meeting latency requirements will have been challenging. Tasks that had been offloaded to other Edge devices cannot be executed due to the lack of resources. As the number of Edge devices grew, the success rates of cases where Edge Data Centers were excluded improved significantly due to the increased number of free Edge devices. In this case, the number of Edge devices with sufficient computability grew, and fewer tasks had to be transmitted to the cloud.

Considering the differences between the different algorithms regarding success rate, there was a slightly favorable trend toward DQN, especially when the number of Edge devices was significant. In this case, there were possible offloading destinations, that is, more possible actions given the state of the environment. By learning an optimal policy, it will be more feasible to reach the optimal download

destination with this algorithm. GNN also outperformed the two default simulator methods when the number of Edge devices was 20 and 30. This was because the graph was complex, and although optimizing the network would be more difficult, the decision to offload was closer to optimal.

In terms of energy efficiency, there was no big difference between the first 3 tests. Obviously, in the case of 20 and 30 devices, the power consumption of Edge devices grew linearly from ~ 3 to ~ 10 Wh and ~ 13 Wh, respectively, due to the larger amount of generated and downloaded tasks to these devices. On the contrary, the energy consumption of the Edge Data Centers almost remained at ~ 34 Wh, even though the number of generated tasks increased. This was because the Edge Data Centers were full and other types of devices were needed to handle the rest of the tasks. This would have caused a reduction in the success rate, especially in the case of the 2 default algorithms and when the cloud and Edge Data Centers were included. In this situation, the rest of the tasks that were not attended to by the Edge Data Centers would have been offloaded to the cloud, meeting the problems mentioned in the previous paragraphs.

Regarding the distribution of tasks between different types of devices, we observed that when the number of Edge devices was not too high (10 or 20 devices), the Edge Data Centers were the destinations for most tasks. In contrast, when the number of devices grew to 30, they did not have enough free memory space, or their processors were busy. As a result, more tasks were offloaded to Edge devices, and a slight increase in the number of tasks offloaded to the cloud was also observed. In the experiment, this study compared only the proposed algorithms since they had the best performance in terms of success rate. Among them, DQN decided to download more tasks to Edge Data Centers, becoming a better alternative due to the better performance when Edge Data Centers were included in the possible download destinations.

Finally, the success rates of different types of devices were carefully compared. This study established all types of devices as possible destinations for the two proposed algorithms. It changed the number of Edge devices to between 10 and 30. There was a clear difference between the performance of the cloud and the rest of the devices. As mentioned in this section, the violation of the latency requirement is responsible for such performance degradation, given the high delay caused when traversing the network to transfer the task to the cloud and return the results to the Edge devices. Between Edge devices and Edge Data Centers, the latter had the best success rate. The larger capacities and larger memory were superior in computability compared to Edge devices. However, with an algorithm good enough to orchestrate all tasks between all possible destinations, offloading the less demanding tasks to the weakest computation centers, the success rate can be preserved with a higher number of generated tasks. That is why the proposed algorithms outperform the two default algorithms: they can offload less demanding tasks to weaker devices and more complex ones to Edge Data Centers. In this way, the task load was balanced between all available devices, meeting latency requirements.

We saw that our proposed algorithms outperform the default PureEdgeSim simulator methods in terms of success rate and load balancing. For example, for the case in which 30 Edge devices generated tasks, GNN and DQN achieved an improvement of 22.1% and 23.8% respectively concerning the Trade-Off when the Edge Datacenters were not included as potential destinations. However, GNN achieved an average improvement of 3.6% concerning Trade-Off and Round Robin, and DQN achieved an average improvement of 4.1% concerning Trade-Off and Round Robin. In other works, such as [53], they achieved an average improvement of 20.48%, 16.28%, and 12.36% concerning random download, higher data rate download (HDR), and the largest computing device (HCD), respectively. In [51], they achieved a 20% reduction in total computation delay and a 25% reduction in average computation delay compared to the GK-means DQN-based offloading policy. In our case, the

rest of the offloading policies analyzed offered a better result since they offered decent behavior in most cases. However, our methods significantly improved the success rates of the mentioned algorithms offer quite similar energy consumption, and have more to do with the distribution of tasks in different layers. Our network environments and experimental setup are completely different compared to those used in the works just mentioned. Therefore, the comparison cannot be made directly between different works. The distribution of tasks was different in our two algorithms, with a larger number of tasks being offloaded to the Edge Data Centers when DQN was applied. This resulted in a slight improvement in the success rate due to the greater capabilities of this type of computing center. Between both types of algorithms, the best results were offered by DQN with a slight variation. The ability to obtain the optimal policy increased when the number of Edge devices and, consequently, the number of generated tasks was larger. The same was true for GNN: by having more nodes and a broader network structure, the algorithm was able to reach a near-optimal offloading decision.

These algorithms could be a useful tool to provide proper orchestration in an environment where many IoT devices are requested to solve complex tasks and the characteristics of the environment are constantly updated. For example, in a Smart Building, several sensors can be located that detect different parameters and have to react by activating any other system based on the readings they obtain. Deciding what action to take may require the use of ML or DL techniques to take the optimal action. In this situation, these small devices could alleviate the computational burden of these deep models by offloading them to other powerful devices such as Edge Data Centers.

In this research, the introduction of GNN and DQN to the paradigm of task-offloading in a local network environment involving IoT, Edge, and Cloud layers is carried out. The similarity between the architectures of a graph and a local network involving the just mentioned devices favored the use of GNN to satisfactorily solve the task offloading paradigm. The offloading ratio used as an edge feature in this study is a good predictor of how good a potential target can be at accomplishing a task. Furthermore, the use of DQN slightly improved the results obtained with GNN. The learning process of the latter favors the consideration of the constant updates of an environment. The novelty offered using the proposed methodology in a local networking environment is the consideration of constant network updates and the scoring of network connections using the novel offloading rating parameter, being both GNNs and DQN powerful tools to impose an optimized offloading strategy in an environment made up of resource-constrained devices.

Among the limitations found during this research work, it is worth highlighting the difficulties in reproducing other methodologies provided by the literature using the PureEdgeSim simulator. The complexity of the simulator was an advantage in adjusting the properties of the network environment to the needs. On the contrary, the reproduction of any algorithm has a high complexity. At the same time, as other environmental properties can directly impact the state of the network, such as vandalism attacks, natural disasters, or intrusion attacks, these must be considered in the simulation, applying a random appearance factor to them.

In future work, more algorithms can be implemented using the simulator to compare them with those presented in this study. Until now, the only default implementable algorithms for the simulator in question were tried and tested against our methods, and due to the complexity of the simulator, no others were implemented. Other types of network structures can be interesting for research and applicable using the methodology proposed in this work. Furthermore, combining RL techniques with the graph will open up an exciting research area.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper. This work is partially supported

by the project a Optimization of Deep Learning algorithms for Edge IoT devices for sensorization and control in Buildings and Infrastructures (EMBED) funded by the Gipuzkoa Provincial Council and approved under the 2023 call of the Guipuzcoan Network of Science, Technology and Innovation Program with File Number 2023-CIEN-000051-01.

Funding Statement: This work has received funding from TECNALIA, Basque Research and Technology Alliance (BRTA). This work is partially supported by the project a Optimization of Deep Learning algorithms for Edge IoT devices for sensorization and control in Buildings and Infrastructures (EMBED) funded by the Gipuzkoa Provincial Council and approved under the 2023 call of the Guipuzcoan Network of Science, Technology and Innovation Program with File Number 2023-CIEN-000051-01.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton; data collection: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton; analysis and interpretation of results: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton; draft manuscript preparation: A. Garmendia-Orbegozo, J. D. Nunez-Gonzalez and M. A. Anton. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data used in the experimental process was generated using the PureEdgeSim simulator, which is perfectly reproducible following the instructions of [Section 4](#).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Yadav, C. S., Pradhan, M. K., Gangadharan, S. M. P., Chaudhary, J. K., Singh, J. et al. (2022). Multi-class pixel certainty active learning model for classification of land cover classes using hyperspectral imagery. *Electronics*, *11*(17), 2799.
2. Haq, M. A. (2022). SmoteDNN: A novel model for air pollution forecasting and AQI classification. *Computers, Materials & Continua*, *71*(1), 1403–1425.
3. You, D., Doan, T. V., Torre, R., Mehrabi, M., Kropp, A. et al. (2019). Fog computing as an enabler for immersive media: Service scenarios and research opportunities. *IEEE Access*, *7*, 65797–65810.
4. ComÅa, I. S., Muntean, G. M., Trestian, R. (2021). An innovative machine-learning-based scheduling solution for improving live UHD video streaming quality in highly dynamic network environments. *IEEE Transactions on Broadcasting*, *67*(1), 212–224.
5. Fraedrich, E., Cyganski, R., Wolf, I., Lenz, B. (2016). User perspectives on autonomous driving a use-case-driven study in Germany. <https://core.ac.uk/download/pdf/31023753.pdf> (accessed on 21/07/2023).
6. Wang, X., Ning, Z., Wang, L. (2018). Offloading in Internet of Vehicles: A fog-enabled real-time traffic management system. *IEEE Transactions on Industrial Informatics*, *14*(10), 4568–4578.
7. Song, D., Tanwani, A. K., Goldberg, K., Siciliano, B. (2019). *Networked-, cloud- and fog-robotics*. Springer. Robotics Goes MOOC, Springer Nature MOOCs, Bruno Siciliano (Editor).
8. Tanwani, A. K., Anand, R., Gonzalez, J. E., Goldberg, K. (2020). RILaaS: Robot inference and learning as a service. *IEEE Robotics and Automation Letters*, *5*(3), 4423–4430.
9. A.W. Services, AWS robomaker (2021). <https://aws.amazon.com/robomaker/> (accessed on 21/07/2023).
10. Google, cloud robotics core (2021). <https://googlecloudrobotics.github.io/core/> (accessed on 21/07/2023).

11. Rapyuta robotics (2021). <https://www.rapyuta-robotics.com> (accessed on 21/07/2023).
12. Papagianni, C., Leivadreas, A., Papavassiliou, S. (2013). A cloud-oriented content delivery network paradigm: Modeling and assessment. *IEEE Transactions on Dependable and Secure Computing*, 10, 287–300.
13. Bilal, K., Erbad, A., Hefeeda, M. (2017). Crowdsourced multi-view live video streaming using cloud computing. *IEEE Access*, 5, 12635–12647.
14. Fajardo, J. O., Taboada, I., Liberal, F. (2015). Improving content delivery efficiency through multi-layer mobile edge adaptation. *IEEE Network*, 29, 40–46.
15. Tran, T., Pandey, P., Hajisami, A., Pompili, D. (2017). Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. *2017 13th Annual Conference on Wireless On-Demand Network Systems and Services (WONS)*. Jackson Hole, Wyoming, USA.
16. Kim, K., Hong, C. S. (2019). Optimal task-UAV-edge matching for computation offloading in UAV assisted mobile edge computing. *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*.
17. Chen, S. C., Lin, C. P., Hsu, H. C., Shu, J. H., Liang, Y. et al. (2019). Serum bilirubin improves the risk predictions of cardiovascular and total death in diabetic patients. *Clinica Chimica Acta*, 488, 1–6.
18. Avgeris, M., Spatharakis, D., Dechouniotis, D., Kalatzis, N., Roussaki, I. et al. (2019). Where there is fire there is smoke: A scalable edge computing framework for early fire detection. *Sensors*, 19(3), 639.
19. Jacob, S., Alagirisamy, M., Menon, V. G., Kumar, B. M., Jhanjhi, N. Z. et al. (2020). An adaptive and flexible brain energized full body exoskeleton with IoT edge for assisting the paralyzed patients. *IEEE Access*, 8, 100721–100731.
20. Farris, I., Militano, L., Nitti, M., Atzori, L., Iera, A. (2016). MIFaaS: A mobile-IoT-federation-as-a-service model for dynamic cooperation of IoT cloud providers. *Future Generation Computer Systems*, 70, 126–137.
21. Lee, E. K., Lewis, D. P. (2006). Integer programming for telecommunications. In: Resende, G. C., Pardalos, P. M. (Eds.), *Handbook of optimization in telecommunications*, pp. 67–102. Boston, MA, USA: Springer. https://doi.org/10.1007/978-0-387-30165-5_3
22. Ketyko, I., Kecskes, L., Nemes, C., Farkas, L. (2016). Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing. *2016 European Conference on Networks and Communications (EuCNC)*. Athens, Greece.
23. Bilal, K., Erbad, A., Hefeeda, M. (2017). Crowdsourced multi-view live video streaming using cloud computing. *IEEE Access*, vol. 5, pp. 12635–12647.
24. Guo, H., Liu, J., Zhang, J. (2018). Computation offloading for multi-access mobile edge computing in ultra-dense networks. *IEEE Communications Magazine*, 56(8), 14–19.
25. Zhao, Y., Hu, W., Yang, Z. (2015). High-resolution transmission electron microscopy study on reversion of al₂cumg precipitates in al-cu-mg alloys under irradiation. *Micron*, 76, 1–5.
26. Chen, X., Jiao, L., Li, W., Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5), 2795–2808.
27. Liu, Y., Xu, C., Zhan, Y., Liu, Z., Guan, J. et al. (2017). Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach. *Computer Networks*, 129, 399–409.
28. Zeng, M., Li, Y., Zhang, K., Waqas, M., Jin, D. (2018). Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach. *2018 IEEE International Conference on Communications (ICC)*. Kansas, MO, USA.
29. Du, J., Gelenbe, E., Jiang, C., Zhang, H., Ren, Y. (2017). Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks. *IEEE Journal on Selected Areas in Communications*, 35(11), 2457–2467.
30. Zhang, Y., Pan, M., Song, L., Dawy, Z., Han, Z. (2017). A survey of contract theory-based incentive mechanism design in wireless networks. *IEEE Wireless Communications*, 24(3), 80–85.

31. Gendreau, M., Potvin, J. Y. (2010). *Handbook of metaheuristics*, vol. 2. Springer.
32. Wang, Y., Breedveld, S., Heijmen, B., Petit, S. (2016). Evaluation of plan quality assurance models for prostate cancer patients based on fully automatically generated pareto-optimal treatment plans. *Physics in Medicine and Biology*, 61, 4268–4282.
33. Yang, B., Cao, X., Basse, J., Li, X., Kroecker, T. et al. (2019). Computation offloading in multi-access edge computing networks: A multi-task learning approach. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. Shanghai, China.
34. Rahbari, D., Nickray, M. (2020). Task offloading in mobile fog computing by classification and regression tree. *Peer-to-Peer Networking and Applications*, 13, 104–122.
35. Hu, R., Jiang, J., Liu, G., Wang, L. (2013). CPU load prediction using support vector regression and Kalman smoother for cloud. *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. Philadelphia, PA, USA.
36. Farahnakian, F., Pahikkala, T., Liljeberg, P., Plosila, J. (2013). Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers. *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. Dresden, Germany.
37. Appadurai, J., Prabaharan, S., Venkateswaran, N., Roseline, S., Rama, B. (2023). Radial basis function networks-based resource-aware offloading video analytics in mobile edge computing. *Wireless Networks*. <https://doi.org/10.1007/s11276-023-03420-7>
38. Cheng, H., Xia, W., Yan, F., Shen, L. (2019). Balanced clustering and joint resources allocation in cooperative fog computing system. *2019 IEEE Global Communications Conference (GLOBECOM)*. Big Island, Hawaii, USA.
39. Li, Y., Anh, N. T., Nooh, A. S., Ra, K., Jo, M. (2018). Dynamic mobile cloudlet clustering for fog computing. *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. Honolulu, Hawaii, USA.
40. Li, G., Lin, Q., Wu, J., Zhang, Y., Yan, J. (2019). Dynamic computation offloading based on graph partitioning in mobile edge computing. *IEEE Access*, 7, 185131–185139.
41. Bouras, I., Aisopos, F., Violos, J., Kousiouris, G., Psychas, A. et al. (2023). Mapping of quality of service requirements to resource demands for IAAS. *Proceedings of the 9th International Conference on Cloud Computing and Services Science CLOSER*, pp. 263–270. Heraklion, Crete, Greece. <https://doi.org/10.5220/0007676902630270>
42. Li, H., Zheng, P., Wang, T., Wang, J., Liu, T. (2022). A multi-objective task offloading based on BBO algorithm under deadline constrain in mobile edge computing. *Cluster Computing*, 26, 4051–4067.
43. Dai, Z., Ding, W., Min, Q., Gu, C., Yao, B. et al. (2023). M-E-AWA: A novel task scheduling approach based on weight vector adaptive updating for fog computing. *Processes*, 11(4), 1053.
44. Hosny, K., Ibrahim Awad, A., Khashaba, M., Rushdy, E. (2023). New improved multi-objective gorilla troops algorithm for dependent tasks offloading problem in multi-access edge computing. *Journal of Grid Computing*, 21, 21.
45. Yu, S., Wang, X., Langar, R. (2017). Computation offloading for mobile edge computing: A deep learning approach. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. Montreal, QC, Canada.
46. Rani, D., Muthukumar, P. (2021). Deep learning based dynamic task offloading in mobile cloudlet environments. *Evolutionary Intelligence*, 14, 499–507.
47. Zhang, R., Cheng, P., Chen, Z., Liu, S., Vucetic, B. et al. (2022). Calibrated bandit learning for decentralized task offloading in ultra-dense networks. *IEEE Transactions on Communications*, 70(4), 2547–2560.
48. Zhang, J., Zhao, L., Yu, K., Min, G., Al-Dubai, A. et al. (2023). A novel federated learning scheme for generative adversarial networks. *IEEE Transactions on Mobile Computing*, 1–17. <https://doi.org/10.1109/ITMC.2023.3278668>

49. Zhang, K., Zhu, Y., Leng, S., He, Y., Maharjan, S. et al. (2019). Deep learning empowered task offloading for mobile edge computing in urban informatics. *IEEE Internet of Things Journal*, 6(5), 7635–7647.
50. Lu, H., Gu, C., Luo, F., Ding, W., Liu, X. (2019). Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Generation Computer Systems*, 102, 847–861.
51. Zhao, L., Zhao, Z., Zhang, E., Hawbani, A., Al-Dubai, A. et al. (2023). A digital twin-assisted intelligent partial offloading approach for vehicular edge computing. *IEEE Journal on Selected Areas in Communications*, 41, 3386–3400.
52. Zhao, L., Zhang, E., Wan, S., Hawbani, A., Al-Dubai, A. et al. (2023). MESON: A mobility-aware dependent task offloading scheme for urban vehicular edge computing. *IEEE Transactions on Mobile Computing*.
53. Maray, M., Mustafa, E., Shuja, J., Bilal, M. (2023). Dependent task offloading with deadline-aware scheduling in mobile edge networks. *Internet of Things*, 23, 100868.
54. Mustafa, E., Shuja, J., Bilal, K., Mustafa, S., Maqsood, T. et al. (2022). Reinforcement learning for intelligent online computation offloading in wireless powered edge networks. *Cluster Computing*, 26, 1053–1062.
55. Liu, J., Wei, X., Wang, T., Wang, J. (2019). An ant colony optimization fuzzy clustering task scheduling algorithm in mobile edge computing. *Security and Privacy in New Computing Environments: Second EAI International Conference, SPNCE 2019*. Tianjin, China, Springer.
56. Hussein, M. K., Mousa, M. H. (2020). Efficient task offloading for IoT-based applications in fog computing using ant colony optimization. *IEEE Access*, 8, 37191–37201.
57. Zhang, D., Haider, F., St-Hilaire, M., Makaya, C. (2019). Model and algorithms for the planning of fog computing networks. *IEEE Internet of Things Journal*, 6(2), 3873–3884.
58. Al-habob, A. A., Dobre, O. A., Garcia Armada, A. (2019). Sequential task scheduling for mobile edge computing using genetic algorithm. *2019 IEEE Globecom Workshops (GC Wkshps)*. Big Island, Hawaii, USA.
59. Li, Y. (2017). *Edge computing-based access network selection for heterogeneous wireless networks (Ph.D. Thesis)*. Université de Rennes 1, France.
60. Avgeris, M., Dechouniotis, D., Athanasopoulos, N., Papavassiliou, S. (2019). Adaptive resource allocation for computation offloading: A control-theoretic approach. *ACM Transactions on Internet Technology*, 19(2), 23. <https://doi.org/10.1145/3284553>
61. Kalatzis, N., Avgeris, M., Dechouniotis, D., Papadakis-Vlachopapadopoulos, K., Roussaki, I. et al. (2018). Edge computing in IoT ecosystems for UAV-enabled early fire detection. *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. Taormina, Sicily, Italy.
62. Pu, L., Chen, X., Xu, J., Fu, X. (2016). D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. *IEEE Journal on Selected Areas in Communications*, 34(12), 3887–3901.
63. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J. et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518, 529–533.
64. Mechalikh, C., Taktak, H., Moussa, F. (2020). PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments. *Computer Science and Information Systems*, 18(1), 43–66.

A.3 Graph Based Learning for Building Prediction in Smart Cities

Received April 7, 2022, accepted April 20, 2022, date of publication April 22, 2022, date of current version May 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169890

Graph Based Learning for Building Prediction in Smart Cities

ASIER GARMENDIA-ORBEGOZO¹, (Member, IEEE), **SARAH NOYE**², (Member, IEEE),
MIGUEL ANGEL ANTON¹, (Senior Member, IEEE),
AND J. DAVID NUÑEZ-GONZALEZ¹, (Senior Member, IEEE)

¹Department of Applied Mathematics, University of the Basque Country, 48940 Leioa, Spain

²TECNALIA, Basque Research and Technology Alliance (BRTA), 20018 Donostia-San Sebastian, Spain

Corresponding author: Asier Garmendia-Orbegozo (asier.garmendiao@ehu.es)

This work was supported by the TECNALIA Member of Basque Research and Tecnology Alliance.

ABSTRACT Anticipating pedestrians' activity is a necessary task for providing a safe and energy efficient environment in an urban area. By locating strategically sensors throughout the city useful information could be obtained. By knowing the average activity of those throughout different days of the week we could identify the typology of the buildings neighboring those sensors. For these type of purposes, clustering methods show great capability forming groups of items that have great similarity intra clusters and dissimilarity inter cluster. Different approaches are made to classify sensors depending on the typology of buildings surrounding them and the mean pedestrians' counts for different time intervals. By this way, sensors could be classified in different groups according to their activation patterns and the environment in which they are located through clustering processes and using graph convolutional networks. This study reveals that there is a close relationship between the activity pattern of the pedestrians' and the type of environment sensors that collect pedestrians' data are located. By this way, institutions could alleviate a great amount of effort needed to ensure safe and energy efficient urban areas, only knowing the typology of buildings of an urban zone.

INDEX TERMS Building prediction, clustering, graph networks, smart city, sensors.

I. INTRODUCTION

The increasing number of Internet of Things (IoT) devices spread throughout cities has evolved in a scenario in which different public services has developed positively, in the way that dynamically information is provided and decision are made in real-time. As a consequence, citizen's lifestyle has become safer, more convenient and environmental issues could be faced up more efficiently.

In a smart city, sensors play the role of collectors, obtaining a huge number of data by sensing different parameters of the environment and different events, such as traffic incidents or pedestrians' mobility. The elevate activity of these devices carries with it a proper maintenance and an intelligent distribution so as to avoid different problems related to safety and energy consumption and to tackle security issues. According to the International Energy Agency (2015), the implementation of a correct control illumination system could save energy by up to 35%. These numbers are behind

the increasing use of more advanced control illumination systems mainly in the commercial and public sectors where lighting represents the highest energy consumption.¹

The positioning of sensors and the distributed management of them is determining. Thus, it is important to make a correct classification of the building typologies, if those could give us the information of the environment they are placed or the pedestrians' tendencies or patterns of mobility.

In this work, which is an extension of "Building typology prediction in Smart Cities" presented in the CIB W78 Information Technology for Construction 39th Conference WBC 2022 [20], two clusterings of sensors were developed. One of them was performed based on the typology of the buildings near-by the sensors, and the second based on the average counts that sensors had made during different time slots throughout different days of the week. After applying Principal Component Analysis (PCA) in order to reduce the

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan ¹.

¹Design of a Smart and Compact Illumination System. Available online at <https://www.redalyc.org/jatsRepo/5722/572261854020/html/index.htmlredalyc.org>

dimensionality of the problem, a clustering was performed and it was created a pair of new datasets adding to each sensor the cluster it belongs to in each case. Later on, supervised Machine Learning (ML) learning algorithms were applied to the latter datasets to validate the clusterings carried out previously.

Finally, a graph convolutional network (GCN) was performed in order to enhance the results obtained from the clustering based on the average counts that sensors had made. By this way, it was feasible to raise different performance metrics of the classification of sensors based on their activity.

This is a novel research in this field due to the lack of attempts to classify sensors based on their characteristics. A similar approach has been carried out in [5] classifying buildings depending on human interaction. In this case, the classification of buildings was done attending the interaction of people and spatio-temporal population density. The main contributions of this work are the following. On the one hand, processing of data to know the activity of sensors and the typology of buildings surrounding them is performed so that sensors are characterized. Next, clustering based on the information obtained in the previous phase is carried out grouping sensors. By this way, pedestrian activity would be predicted giving the opportunity to anticipate in different ways providing an eco-friendly and safe urban environment. Finally, an alternative approach driven by a GCN is carried out.

The remainder of this work is organized as follows. Section 2 reviews the literature. In section 3 the fundamental concepts about PCA, clustering and supervised machine learning are described and the proposed methodology is reviewed. The materials used in this work are described in Section 4. The experimental work is presented in Section 5. The experimental results and analysis are presented in Section 6. Conclusions of this work and outlines of some potential directions for further investigation are made in Section 7. The abbreviations cited in this paper are summarized in Table 12.

II. LITERATURE REVIEW

In the way of achieving a smart distribution and maintenance of cities different researches have been made recently in order to tackle a wide variety of issues, such as building functionality identification, pedestrians detection, traffic prediction or other type of detections. Different skills have been used in those works enabling optimum solutions to the mentioned tasks.

Different techniques have been developed to address tasks related with traffic. In [2] a camera based system was used, even though bad visibility conditions caused by bad weather or insufficient lighting were limiting factors. The same problem was limiting their performance in infrared sensor based system [3]. Similarly, in [4] a sparse coverage of video cameras in the public space was performed, acceptance levels amongst citizens being too low, though.

Identifying the buildings' typology has been useful in a wide range of applications. In [5] a new method to identify building functions from the perspective of the spatial distribution and spatial interactions of human activities was proposed. First, taxi data were used to acquire the spatiotemporal interaction characteristics among buildings with different functions. Then, the spatiotemporal population density distribution was adopted to depict the building vitality. Finally, an iterative clustering method was introduced to identify the building functions.

The correlation between space and time in smart cities has emerged the need of different research lines so as to solve the lack of works bridging this two variables. Heterogeneity related to the distribution within space and the dynamism of data through time has led to the partitioning of space in regions and time intervals. In [9] they provide a method for combining both spatial and temporal factors in predicting pedestrian flow within city centres. The model utilizes sensor data over an extended time period that allows seasonality and time of day factors to be incorporated as well as actual walking distances to points of interest and transport terminals in contrast to Euclidean distances to identify influencing factors.

Different approaches has been made in order to solve issues related to traffic incidents, by using binomial logistic regression and space-time cube model [6] or geographic information system (GIS) to visualize the distribution of pedestrian crashes in cities to explore the relationships between pedestrian crashes and the population, road network, land use and social services and activities and to analyze the impacts of the building environment and road characteristics on the severity of pedestrian crashes by combining the binary logistic regression and tree-based models [7], among others. Nowadays, the complex spatial dependency of road networks, non-linear temporal dynamics with changing road conditions, and the inherent difficulty of long-term forecasting is challenging. In [8] there is a presented deep learning framework titled, "Diffusion Convolutional Recurrent Neural Network (DCRNN)" for traffic forecasting. With the aim of making traffic prediction and the incorporation of spatial and temporal dependency in the traffic flow, DCRNN also integrates the encoder-decoder architecture with a scheduled sampling technique to improve performance and long-term forecast of traffic.

With the advancement of AI techniques, new methods have emerged to perform clustering tasks. In that sense, graph clustering has become one of the most popular and widely adopted methods. There are an increasing number of applications that use graphs to represent data. For example, in e-commerce, a graph-based learning system can provide extremely accurate suggestions by using the interactions between customers and products, recommender system, social networks, biological protein-protein networks [10]. In chemistry, molecules are represented as graphs, and their bioactivity must be determined in order to develop new drugs. Whereas citation network; traffic forecasting, taxi demand

prediction are all used to predict the concentrations of a wide variability of air pollutants. By forecasting the crowd flows to predict urban traffic flow, management of tourism flows can be predicted. Finding the way to incorporate graph structure information into a machine learning model is the core problem in graph machine learning.

Traffic forecasting is essential for guidance and traffic control. In [11] there is a proposed model based on Spatio-Temporal Graph Convolutional Networks (STGCN) for traffic prediction. STGCN seeks to predict in the traffic domain by integrating convolution with graphs and space-time convolution in blocks so that the training is faster with a smaller number of parameters. In [12] a proposed Origin – Destination based Temporal Graph Attention Network (OD-TGAT) framework is used for taxi demand forecasting. This model has two main building blocks: a graph network and a neural network. This is the first representation of a model employed graphing network used for taxi demand prediction. In [13] there is a proposed hybrid model based on deep learning methods. This hybrid model integrates Graph Convolutional networks and Long Short-Term Memory networks (GC-LSTM) to establish and predict the spatiotemporal variation in space-time of $PM_{2.5}$ concentrations by applying the graph convolutional networks (GCN) to extract the spatial dependency between different stations, as well as the Long Short-Term Memory (LSTM) to capture the temporal dependency between observations at different times.

Forecasting the crowd flows in each and every part of a city, especially in irregular regions, is very important for the following reasons: traffic control, risk assessment, and public safety. Nevertheless, it is very challenging because of the interactions and spatial correlations between different regions. In [14], the proposed multi-view graph convolutional network (MVGCN) is used to predict the inflow and outflow in each and every irregular region of a city to integrate the geospatial position via spatial graph convolutions. In [15] the proposed attention-based deep spatio-temporal network, with multi-task learning (ADST-Net) at a citywide level, creates a goal to predict urban traffic flow. ADST-Net furthermore introduces an outside embedding mechanism to extricate the impact of external factors on flow prediction, such as weather conditions.

III. FUNDAMENTAL CONCEPTS AND PROPOSED METHODOLOGY

A. PROPOSED METHODOLOGY

This research has followed the approach described in this section. As it is shown in Fig. 1 one can distinguish 3 main phases. The second one, could be divided into 2 subphases depending the architecture used to tackle the problem in question. Both of them, the GCN and the clustering method are based on the previous processing of data obtained from open data source from the city of Melbourne. In this first part of the research new datasets derived from the ones

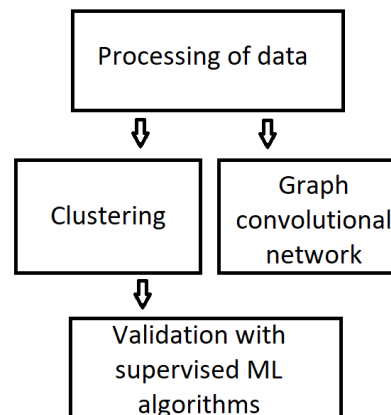


FIGURE 1. Diagram of the proposed methodology.

obtained from the open sources were calculated followed by a dimensionality reduction technique. The three dimensionality reduction techniques that we compared were Principal Component Analysis (PCA), Unifold Manifold Approximation and Projection (UMAP) and t-Distributed Stochastic Neighbor Embedding (tSNE). The activity throughout different time intervals of days of the sensors was calculated as well as the number of buildings of each type of their surroundings as explained in section IV, where materials are described in depth. After this, reduction techniques were carried out to reduce the dimensionality of the issue, and use fewer variables in the next phase raising the efficiency of these.

In the second phase with the information obtained in the previous one a clustering process or a graph convolutional network is carried out to distinguish groups of sensors depending on their activity throughout the day or the type of environment (building) they are located. In case of the clustering processes a posterior validation using supervised Machine Learning algorithms was performed to show the effectiveness of the previous approach. For both datasets, 3 clusterings were done to determine which of the dimensionality reduction techniques suits best for this problem. After all, we verified that PCA was the most adequate method to use in this case, continuing the rest of the work applying this technique.

Within this section a more detailed explanation of each of the concepts of the phases mentioned above is given.

B. PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis is the process of computing the principal components of a collection of points, that are sequence of p unit vectors where the i -th vector is the direction of a line that best fits the data while being orthogonal to the first $i-1$ vectors, and using them to perform a change of basis on the data. One of the objectives of using this method is to carry out the dimensionality-reduction of a data set with a large number of connected variables while

maintaining as much variance as feasible in the data set. This is accomplished by converting to a new set of uncorrelated variables known as principle components (PCs), which are sorted so that the first few keep the majority of the variance existing in a dataset [16]. Among other dimensionality reduction techniques PCA offers lowest computational cost compared to tSNE or UMAP. To decide which of them fits best our problem, we developed part of the methodology with each of the 3 dimensionality reduction techniques and after all, we saw that clustering of sensors provides a better accuracy with PCA than with the other two methods. In fact, PCA outperformed in a range of 5% the performance of UMAP and in 10% the performance of tSME. Further details are contained in section 6. Consequently, during the rest of the work we adopted PCA as the dimensionality reduction technique.

Generally, a reduction in the number of variables in a data set carries a reduction in accuracy. Nevertheless, there is a trade-off between accuracy and simplicity. The reason for this reduction is not only that smaller data sets are easier to study and display, but also the machine learning algorithms can analyze data more easily and quickly without having to deal with superfluous factors.²

In this case, adopting this technique is the optimum solution in order to lower the number of dimensions of the problem in question.

C. CLUSTERING

Clustering is a type of unsupervised learning method. Unsupervised learning is a technique for extracting references from datasets that contain input data but no labelled answers, and self-discovering naturally occurring patterns. It is a method for identifying significant structure, explaining underlying processes, generating traits, and groups in a set of samples.

Clustering is the process of partitioning a population or set of data points into several groups so that the similarity of points within a group is high and dissimilarity between points from different groups is high, as well. It is essentially a grouping of items based on their similarity and dissimilarity.

This method is critical since it determines the inherent grouping among the unlabeled data. There are no requirements for a successful clustering. It is up to the user to determine what criteria employ to satisfy its needs. For instance, we might be interested in locating representations for homogeneous groups (data reduction), locating “natural clusters” and describing their unknown qualities (“natural” data types), locating useful and appropriate groupings (“useful” data classes), or locating odd data objects (outlier detection).

It is worth differentiating between fuzzy clustering and hard/crisp clustering. The former one gives the degree of

belonging to each cluster for each item, whereas the latter one classifies each item to an unique cluster.

Attending the criteria used for dividing the clusters by the algorithm, different clustering methods can be defined. These are the type of methods and the most important examples of them:

- Density-Based Methods: k-Means, Partitioning Around Medoids (PAM), Clustering Large Applications (CLARA), k-Prototypes, K-Mode.
- Hierarchical Based Methods: Sequential Agglomerative Hierarchical Non-overlapping (SAHN), Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH), Clustering Using Representatives (CURE), Robust Clustering using Links (ROCK).
- Partitioning Methods: Density-based spatial clustering of applications with noise (DBSCAN), Density-based Clustering (DENCLUE).
- Grid-based Methods: Statistical Information Grid (STING), Wavecluster.

The simplest and most satisfactory unsupervised machine learning approach for solving the clustering problem in many cases is the K-means clustering algorithm. The K-means algorithm divides n observations into k clusters, with each observation belonging to a cluster. The centroids of each cluster are initialized randomly from the initial observation set, and the rest of the items are assigned to the nearest centroid's cluster. After each assignation the centroids are recalculated and identical process is repeated until there are no remaining observations to classify.³

D. SUPERVISED MACHINE LEARNING METHODS

The supervised machine learning techniques aim to classify a set of items based on their features and other set of pre-classified items with the same features. This techniques infer a function from a training data set to use it to classify other instances from a test data set. Each instance of a training set consists of a set of features seen as an input vector and the desired output, which is the remaining feature for the instances of the testing data set.

After differentiating those two data sets, the inferred function is used for predicting the output for the instances of the test data set (known beforehand). By this way, the accuracy of the algorithm could be stated comparing the results obtained from the classification process and the ground-truth. As well as that, those algorithms could be used for validating the results of a clustering process.

There are many algorithms that can address this task, and variations of them can be found in the literature. These are some of the most commonly used ones: Naive-Bayes, Decision Tree, Supported Vector Machine, Artificial Neural Networks, Boosting methods, Bagging methods, etc.

²A Step-by-Step Explanation of Principal Component Analysis (PCA). Available online at <https://builtin.com/data-science/step-step-explanation-principal-component-analysisbuiltin.com>

³Clustering in Machine Learning. Available online at <https://www.geeksforgeeks.org/clustering-in-machine-learning/geeksforgeeks.org>

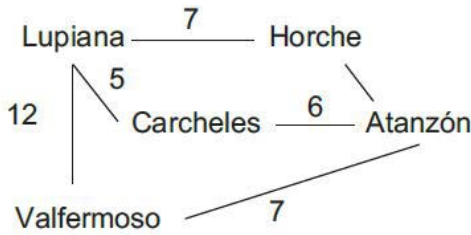


FIGURE 2. Valued undirected graph.

E. BASICS OF NEURAL NETWORKS

An artificial neural network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. These values can be integer, real, or binary. Based on the inputs and the weights, the weighting function produces a weighted sum that is passed through an activation function to produce an output.

F. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

The higher performance of convolutional neural networks or ConvNet with picture, speech, or audio signal inputs sets them apart from other artificial neural networks [18].

They are divided into three sorts of layers:

- Convolutional layer. The convolutional layer is the central component of a CNN, and it is here where the majority of the computation takes place. Input data, a filter, and a feature map are components required.
- Pooling layer. Downsampling, also known as pooling layers, is a dimensionality reduction technique that reduces the number of factors in the input. The pooling process sweeps a filter across the entire input, similar to the convolutional layer, however this filter does not have any weights.
- Fully-connected (FC) layer. The full-connected layer's name is self-explanatory. In partially linked layers, the pixel values of the input image are not directly connected to the output layer, as previously stated.

G. GRAPH THEORY

1) BASIC CONCEPTS

A graph is represented by the pair $G = (V, A)$. The V represents the set of vertices or nodes and the A the set of edges (or arcs).

The nodes represent the elements of the system and edges the interrelationships between them. If all the edges can be traversed in both directions, the graph is known as undirected. In the case of directed graphs, each edge has a direction, generally represented by its origin node and its destination node.

Valued graph: is a graph G together with a function W_E that assigns a numerical weight W_{ij} to each edge (i, j) . Eventually it can also coexist with a W_V function that assigns a W_i value to each i node.

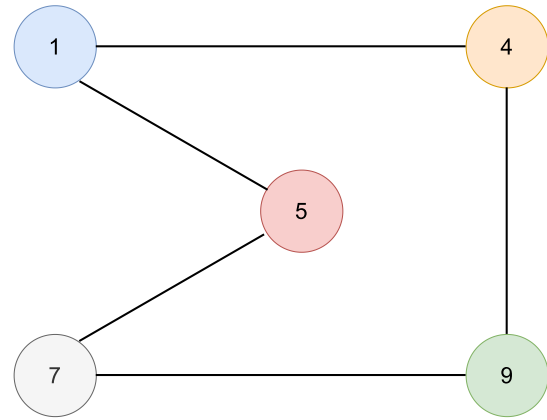


FIGURE 3. Undirected graph.

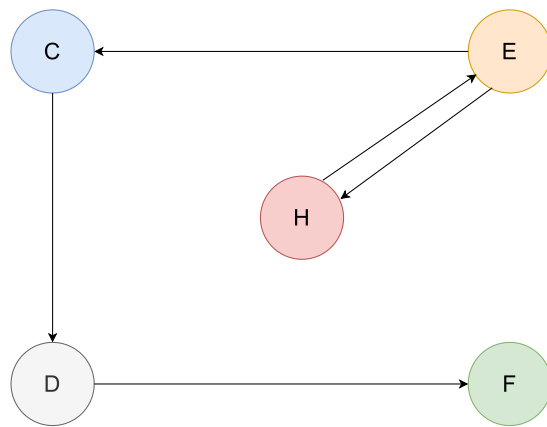


FIGURE 4. Directed graph.

As can be seen in the Fig. 2 a valued graph is shown in which each arc has an associated weight that is the length between two nodes [19].

The graph is undirected if the arcs are formed by pairs of unordered vertices, not pointed.

As can be seen in the Fig. 3 an undirected graph is shown formed by the vertices $V = \{1, 4, 5, 7, 9\}$ and the set of arcs $A = \{(1,4), (4,1), (5,1), (1, 5), (7,9), (9,7), (7,5), (5,7), (4,9), (9,4)\}$.

When a graph is directed, it is also known as diagraph. In this type of graph the pairs of nodes that form the edges are ordered and are represented by an arrow indicating the direction of the relationship $u \rightarrow v$.

As can be seen in the Fig. 4 a directed graph is shown formed by the vertices $V = C, D, E, F, H$, and the arcs $A = \{(C, D,), (D, F), (E, H), (H, E), (E, C)\}$ form the directed graph $G = V, A$.

2) GRAPH REPRESENTATION

- List of neighbors: associates to each node the list of its neighbors, that is, $neighbors(i) = j : (i, j) \in E$
- Adjacency matrix: of Boolean values 0, 1 such that $M(i, j) = 1 \Leftrightarrow (i, j) \in E$

The graphs can be applied to different tasks such as the following:

- Node Classification: this categorization, which is founded exclusively on non-attribute graphs, is based on the graph's structure and the class of the known nodes in our experiment.
- Link prediction: here the nodes' classes aren't taken into account.
- Node clustering: node clustering can be applied to a group items by their proximity to each other.

H. GRAPH NEURAL NETWORKS (GNNS) AND GRAPH CONVOLUTIONAL NETWORKS(GCNS)

Graphs are a type of data structure that represents a collection of items (nodes) and their connections (edges). A Graph Neural Network is a sort of Neural Network that works with the graph structure directly. Node categorization is a common use of GNN. Every node in the network has a label, and predictions of the labels of the nodes using ground-truth data is being made. Convolutional networks multiply the input neurons with a set of weights that are commonly known as filters or kernels. The filters act as a sliding window across the whole image and enable CNNs to learn features from neighboring cells. GCNs perform similar operations where the model learns the features by inspecting neighboring nodes. The major difference between CNNs and GNNs is that CNNs are specially built to operate on regular (Euclidean) structured data, while GNNs are the generalized version of CNNs where the numbers of nodes connections vary and the nodes are unordered (irregular on non-Euclidean structured data).⁴

In our experiment, sensors represent the nodes of the graph, so that node classification is performed to obtain different sensor groups. These nodes' features represent the mean activity of sensors. The number of features was reduced by applying PCA, finally obtaining only 5 features per node for the clustering based in the activity of sensors, and 14 features for the clustering based in the buildings' typology. On the other hand, edges are the invert of the relative distances between sensors.

IV. MATERIALS AND METHODS

In this section, we describe the materials used in this work. We introduce a description of the datasets used.

A. DATASETS

The data that has been used in this work is partly from the city of Melbourne.⁵ In this portal we can find an open dataset: Pedestrian Counting System - Monthly (counts per hour). This dataset contains hourly pedestrian counts since

⁴Understanding Graph Convolutional Networks for Node Classification. Available online at <https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7btowardsscience.com>

⁵City of Melbourne Open Data, available at [https://data.melbourne.vic.gov.au/stories/s/data-principles/9f8u-v2fn?src=hdrCity of Melbourne Open Data](https://data.melbourne.vic.gov.au/stories/s/data-principles/9f8u-v2fn?src=hdrCity%20of%20Melbourne%20Open%20Data)

2009 from pedestrian sensor devices located across the city. In the same way we can find Pedestrian Counting System - Sensor Locations. This dataset contains status, location and directional information for each pedestrian sensor device installed throughout the city transportation. Finally, Buildings with name dataset contains the information about the typology of the buildings and their locations.

In order to expand the amount of data used to train the classifiers described in section V we made use of the pedestrian mobility information provided by the city hall of Madrid.⁶ There were counts of pedestrian and cyclists from 2019 to 2021. Although there were counts available of 2019, due to several break downs of sensors there were various missing values. Furthermore, we considered only data between 2020 and 2021.

1) PEDESTRIAN COUNTING DATASETS OF MELBOURNE

The Pedestrian Counting System - Monthly (counts per hour) dataset⁷ contains hourly pedestrian counts (since 2009) using pedestrian sensor devices located across the city. The data is updated on a monthly basis. This dataset contains 3,482,938 records in all, and it has been collected from May 1st, 2009 to December 31th, 2020.

In order to avoid analysing instances from sensors that were not working or were damaged, we used another dataset named Pedestrian Counting System - Sensor Locations dataset.⁸ This dataset contains information about status, location and direction for each pedestrian sensor device installed throughout the city. It is made available by the City of Melbourne with a Creative Commons Attribution 4.0 International license.⁹

We conducted to the exploratory analysis of the data from the previous data set in order to know if all the sensors have enough information to include them later in a PCA analysis and clustering. It is important to know the activity of the sensors and see in which period there was no record of pedestrian crossings as part of the exploratory analysis. After this, we removed the instances from sensors which had not been operating.

After all, we decided to establish two hour time intervals for each day and separate weekdays from Saturdays and Sundays. By this way, we generated a new dataset (named Melbourne pedestrians with mean hourly count datafinal all and both hourly counts, for both cities' mean values) that

⁶City of Madrid Open Data, available at [https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9f8e4b2e4b284f1a5a0/?vgnnextoid=695cd64d6f9b9610VgnVCM1000001d4a900aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default#City of Madrid Open Data](https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9f8e4b2e4b284f1a5a0/?vgnnextoid=695cd64d6f9b9610VgnVCM1000001d4a900aRCRD&vgnnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnnextfmt=default#City%20of%20Madrid%20Open%20Data)

⁷Pedestrian Counting System - Monthly (counts per hour). Available online at https://data.melbourne.vic.gov.au/Transport/Pedestrian-Counting-System-Monthly-counts-per-hour/b2ak-trbp?src=featured_bannerdata.melbourne.vic.gov.au

⁸Pedestrian Counting System - Sensor Locations. Available online at <https://data.melbourne.vic.gov.au/Transport/Pedestrian-Counting-System-Sensor-Locations/h57g-5234data.melbourne.vic.gov.au>

⁹Creative Commons Attribution 4.0 International license, available at <https://creativecommons.org/licenses/by/4.0/legalcodecreativecommons.org>

included the mean hourly count value for each sensor in different time intervals of different days (weekday, Saturday, Sunday).

2) BUILDINGS WITH NAME DATASET OF MELBOURNE

This dataset contains the typology of buildings that will be used to calculate the geodesic distance that exists from a pedestrian sensor to a type of building. By this way, we registered the buildings that were under a previously defined neighboring distance (300 m) from each sensor (obtaining their location from Pedestrian Counting System - Sensor Locations dataset) and these values were used later to perform a PCA and posterior clustering of sensors depending on the typology of the buildings near-by. Thus, the predominant types of buildings surrounding each sensor could be known. These results were summarised in a new dataset (named Melbourne count pedestrians buildings) that had been used in the posterior phases.

3) MADRID PEDESTRIAN DATASETS

These datasets contain the hourly pedestrian counts of every sensor spread throughout the city of Madrid for each year. They contain information about each sensor, its latitude and longitude, information of its address, and the typology of the address (pedestrian street, sidewalk, etc.).

V. EXPERIMENTAL WORK

After obtaining the desired data, we sought out to divide the sensor into different groups or clusters. For this purpose it is advisable to reduce the number of features of each dataset, not only for enhancing the interpretability of the data, but also for improving the results of clustering, because the machine learning algorithms can analyze data more easily and quickly without having to deal with superfluous factors. After this reduction had been made, we present the techniques used to divide the sensors based on different criteria, depending on the building typology surrounding them and the number of the activations that they had in different time intervals throughout the week. Finally, we validate each division made in the previous phase by using supervised machine learning algorithms.

A. PRINCIPAL COMPONENT ANALYSIS (PCA)

First, we analysed the cumulative explained variance of different numbers of components for each dataset, after scaling the data with mean 0 and standard deviation of 1 for each attribute for optimizing the results. There is no single answer or method to identify the optimal number of main components to use. A very widespread way of proceeding consists of evaluating the proportion of accumulated explained variance and selecting the minimum number of components from which the increase is no longer substantial. Depending on the degree of accuracy required this proportion varies. In our case, we established a 95 % of variation to be explained by the amount of components selected.

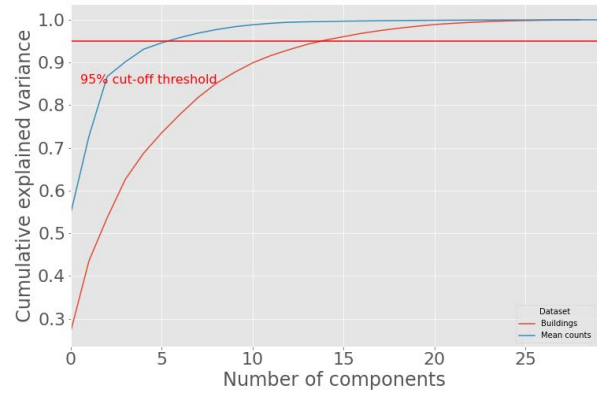


FIGURE 5. Cumulative explained variance for the Melbourne count pedestrians buildings and Melbourne pedestrians with mean hourly count dataset.

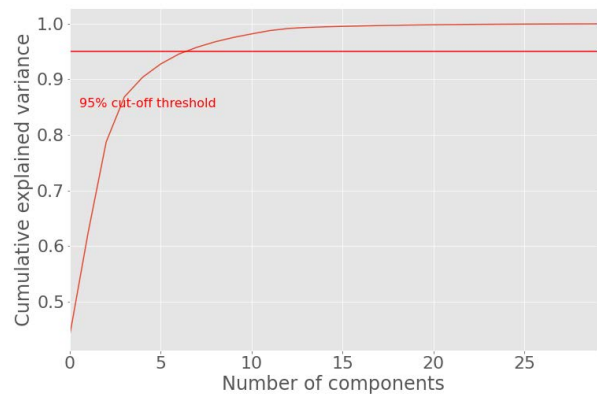


FIGURE 6. Cumulative explained variance for the Melbourne & Madrid mean hourly count dataset.

As it can be seen in the Fig. 5, with 14 components the desired variance is achieved for the Melbourne count pedestrians buildings dataset. In the same way, 5 components were sufficient to satisfy the requirement mentioned above for the Melbourne pedestrians with mean hourly count dataset. However, 6 was the optimum number of components to satisfy the mentioned requirement in case of both cities' pedestrians activity pattern dataset, as it can be seen in Fig. 6.

B. CLUSTERING

1) SELECTING OPTIMAL NUMBER OF CLUSTERS

To select the optimal number of clusters there are different traditional methods. For the problem in question, we are going to use the elbow method. The function Inertia simply computes the squared distance of each sample in a cluster to its cluster center and sums them up. The smaller the Inertia value, the more coherent are the different clusters. When as many clusters are added as there are samples in the data set, then the Inertia value would be zero. After obtaining principal components, we proceed to apply the K-Means algorithm after evaluating the optimal number of clusters according to the elbow method. The point in which the graph flattens will indicate the optimal number of clusters, just enough to achieve a desired difference and coherence between clusters.

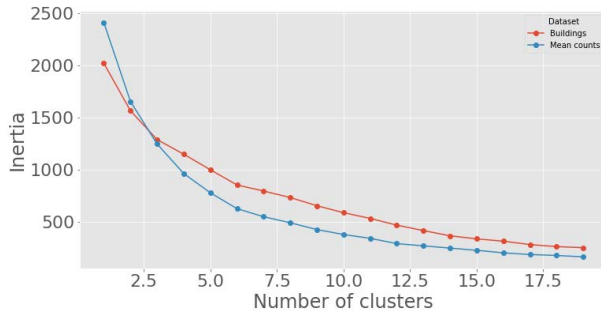


FIGURE 7. Elbow method to obtain the optimal number of clusters for Melbourne pedestrians with mean hourly count datafinal all dataset.

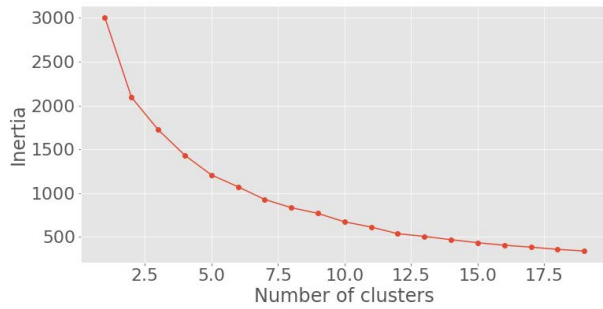


FIGURE 8. Elbow method to obtain the optimal number of clusters for Melbourne & Madrid pedestrians with mean hourly count dataset.

TABLE 1. Sensors and clusters based on buildings’ typology.

Cluster Name	Sensors
Office	45, 63, 66, 15, 47, 3, 21, 30, 56, 52, 2, 1, 20, 19
Rest of sensors	7, 14, 25, 64, 17, 29, 23, 33, 13, 12, 9, 6, 42, 5, 35, 34, 57, 40, 58, 8, 11, 18, 24, 28, 44, 10, 43, 73, 75
Residential Apartment	55, 61, 62, 54, 46, 59, 49, 26, 51
House-Community use	70, 27, 48
Office-Retail	72, 41, 60, 39, 65, 67, 68, 69, 71, 13, 53, 38, 32, 4, 50, 40, 31, 37, 36, 16, 22

As it can be seen in the Fig. 7, 6 clusters would be a reasonable option for the Melbourne pedestrians with mean hourly count datafinal all dataset, but 4 also could be considered. At the beginning we decided to choose 6 clusters and similarly for the Melbourne count pedestrians buildings dataset, with 6 clusters as well.

When considering data from both cities we concluded that 5 would be the optimum number of clusters as it is shown in Fig. 8

2) CLUSTERING BASED ON BUILDINGS’ TYPOLOGY

The Fig. 9 shows the clusters based on the typology of buildings. We can differ some clear groups analysing only the two first principal components. But as we will see in the section 6 the fusion of the clusters 4 and 6 improved the supervised classifiers performance.

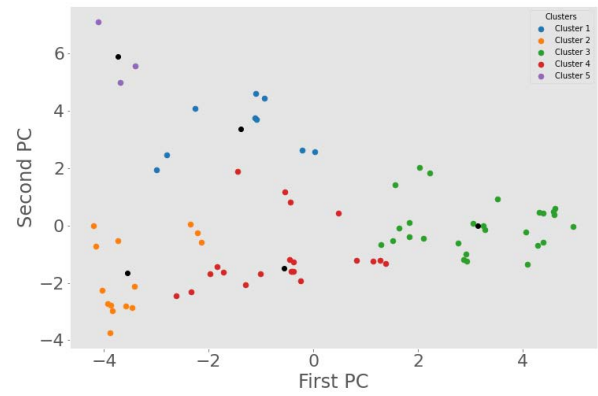


FIGURE 9. Clustering based on the typology of buildings. Black points indicate the centroids of each cluster.

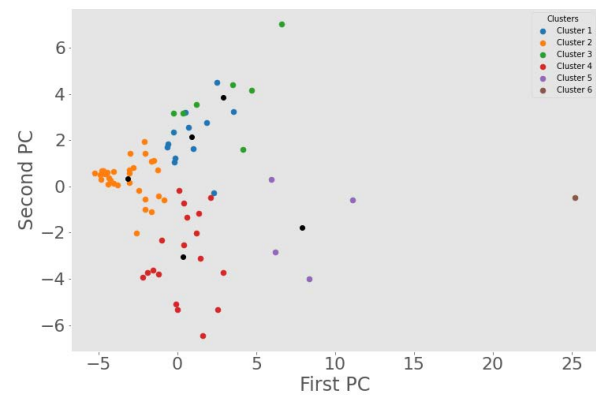


FIGURE 10. Clusters based on pedestrians’ activity. Black points indicate the centroids of each cluster.

Table 1 shows the cluster’s name based on the main type of buildings that have near-by the sensors included on it. The sensors of the cluster named “Rest of sensors” do not have a clear predominance of typology of any building. Thus, was made its nominalisation.

3) CLUSTERING BASED ON THE ACTIVATION OF THE SENSORS

In the clustering based on the mean activations of the sensor per time interval throughout the week we obtained the clusters shown in the Fig. 10.

Clearly, there is a sensor that differs completely from the rest observing the two first principal components. In the results and analysis section (section 6) we could see that taking out this data the latter classifier outperformed the former one. At the same time, it is noticeable the similarity between the clusters 1 and 3, and for this reason we saw that the accuracy of the classifier after fusing clusters 1 and 3 was enhanced.

Table 2 shows the name of the clusters and sensors based on their activity. Depending whether their sensors’ main activations were concentrated in weekdays or weekends and their frequency of activations was the nominalisation of clusters made. The sensors that have a very high activity pattern

TABLE 2. Sensors and clusters based on the pedestrians’ activity of Melbourne.

Cluster Name	Sensors
Weekday	39, 37, 40, 43, 44, 48, 56, 17, 53, 10, 12, 11, 14, 26, 23, 25, 19, 27, 31, 33, 63, 64, 65, 66, 67, 68, 69, 70, 71, 73
Weekend+	29, 46, 50, 51, 62, 8, 30, 34, 36, 42, 43, 49, 52, 54, 56, 57, 59, 61, 18, 7, 20
Weekday++	35, 45, 47, 55, 58, 2, 1, 3, 15, 9, 5, 6, 24, 22, 21, 28, 13, 16
Weekday-weekend++	4, 41, 32, 60

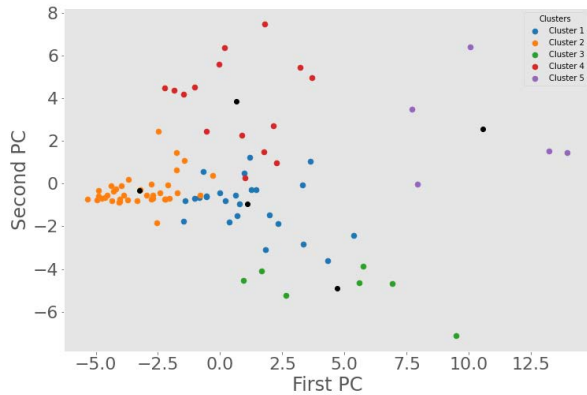


FIGURE 11. Clusters based on pedestrians’ activity from Madrid & Melbourne. Black points indicate the centroids of each cluster.

TABLE 3. Sensors and clusters based on the pedestrians’ activity of Melbourne & Madrid.

Cluster Name	Sensors
Weekend	34, 40, 36, 42, 43, 49, 52, 54, 56, 57, 58, 59, 61, 17, 18, 9, 7, 20, 27, 16, 66, 77, 78, 85, 88
Weekday	39, 37, 44, 48, 53, 10, 12, 11, 14, 26, 23, 25, 19, 31, 33, 63, 64, 65, 67, 68, 69, 70, 71, 73, 74, 75, 76, 77, 78, 81, 82, 83, 84, 85, 86, 88, 89, 91
Weekend+	5, 11, 15, 16, 25, 38, 54
Weekday+	9, 10, 12, 19, 30, 31, 32, 33, 40, 41, 44, 47, 50, 52, 56
Weekday-weekend+	26, 35, 58, 60, 71

on weekdays were grouped in weekday++ cluster, while the ones that were mostly activated in weekdays but with a minor activity were grouped in weekday cluster. Sensors whose activity was concentrated in weekends were grouped in weekend cluster and the ones that follow a similar activity pattern throughout all the week were grouped in weekday-weekend++ cluster, whose activity was also very high.

In order to conclude more accurate results, as we mentioned above we merged pedestrians’ activity data from Madrid and Melbourne. We repeated the process of clustering and we obtained the clusters shown in Fig. 11.

C. VALIDATION USING SUPERVISED ML ALGORITHMS

Next step was to validate the goodness of the clusterings performed in the previous phase by applying supervised machine

TABLE 4. The optimal parameter values for different machine learning algorithms.

Algorithm	Param. 1	Param. 2	Param. 3	Param. 4
SVM	kernel='poly' ['linear', 'poly', 'rbf', 'sigmoid', 'precomputed']	degree of polynomial function=2 [2,3,4,5,6]	kernel gamma coefficient='auto' ('scale', 'auto')	
RF	n estimators= 130 [30, 250]	criterion='entropy' ['gini', 'entropy']	max depth=4 [2, 7]	
DT	criterion = 'entropy' ['entropy', 'gini']	min samples to split a node = 5 [2,7]	min samples per leaf = 3 [1,6]	max depth = 10 [2,20]
MLP	hidden layer sizes = 250 [30,3000]	max iterations = 700 [150,2000]		
Adaboost	base estimator = RF [RF, SVM, GNB]	n estimators = 10 [5,20]		
Bagging	base estimator = RF [RF, SVM, GNB]	n estimators = 10 [5,20]		

learning algorithms in both cases. With the obtained clusters we modified the former datasets and included the cluster each sensor belongs to as an extra feature, that was to be used as the output of the machine learning algorithms. By this way, observing different performance metrics of the algorithms we can deduce the goodness of the clusterings.

For all datasets different machine learning algorithms were applied. Those were: Supported Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), Multilayer Perceptron (MLP), Gaussian Naive-Bayes (GNB), Adaboost (base classifier: RF for the Buildings’ dataset and DT for the mean activation’s dataset) and Bagging (base classifier: SVM). For them, an optimization of the parameters was done, in order to enhance their performance. The optimal hyperparameters and the range of parameters used to find the optimal ones (between parentheses) are given in Table 4.

1) MODEL EVALUATION METRICS

The performance of our model was evaluated using the following metrics:

- Confusion matrix: It is a specific table structure that shows the performance of an algorithm class by class. The examples of an actual class are represented by each row of the matrix, whereas the instances in a predicted class are represented by each column.¹⁰

¹⁰Confusion matrix, Available online at https://en.wikipedia.org/wiki/Confusion_matrix

- Accuracy: It is the proportion of correct predictions among the total number of cases being examined.

$$Accuracy(Acc) = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- Precision: It gives the probability of an instance predicted of one class being of such class in reality.

$$Precision(Pr) = \frac{TP}{TP + FP} \quad (2)$$

- Recall: It gives the probability of an instance belonging to a class being predicted of such class.

$$Recall(Re) = \frac{TP}{TP + FN} \quad (3)$$

- F-Score: A performance metric that takes into account the trade-off between Precision and Recall.

$$F - Score(F) = \frac{2 * Pr * Re}{Pr + Re} \quad (4)$$

Notation: TP = True positive; FP = False positive; TN = True negative; FN = False negative

D. GRAPH CONVOLUTIONAL NETWORK

After obtaining a ground-truth data from the clustering based on the activity patterns of sensors, this was used to build the convolutional graph. The nodes of the graph were the sensors themselves and the edges were weighed by the invert of the distance used as their feature. Those all distances were obtained from the Pedestrian Counting System - Sensor Locations dataset.

We performed a two convolutional layer network. The first layer has as input features the number of components(5) and 22 features as output. The second layer has these 22 features as input and the number of classes as output features. The number of features was adjusted manually to optimize the accuracy of the network. Simpler layers could not achieve the desired performance, and with more features the structure was too complex for the learning process of this problem. The activation function used was Relu. For training the network 0.001 learning rate was used, and the optimization algorithm chosen was Adam. These last parameters were adjusted manually to reach the highest performance of the network as well. A too low learning rate derives in a computationally too long learning process, while a higher one would not produce optimal results.

The feature matrix was calculated using the components obtained in PCA for the Melbourne pedestrians with mean hourly count data in all dataset derived from Pedestrian Counting Datasets of Melbourne(2009-2020). The edges of the graph were the inverse of relative distances between sensors (nodes), thus strengthening the links between proximal sensors.

The division of the sensor belonging to each cluster was done in the following way: 80% was used for training process, 10% was used for validation phase and the rest 10% for

TABLE 5. Accuracies for building typology (B) and time-interval mean activation (A) datasets based clustering after 3 different dimensionality reduction techniques.

Classifier	B(PCA)	B(tSNE)	B(UMAP)	A(PCA)	A(tSNE)	A(UMAP)
SVM	93.39%	81.25%	85.36%	87.14%	70.54%	59.46%
RF	94.14%	84.88%	89.77%	85.07%	78.99%	82.90%
DT	88.86%	68.73%	71.54%	87.43%	61.33%	59.92%
MLP	81.86%	79.23%	80.56%	66.71%	57.33%	63.08%
GNB	91.96%	71.61%	86.96%	85.71%	82.14%	80.54%
Adaboost	95.15%	85.24%	89.52%	81.85%	77.17%	79.70%
Bagging	91.54%	80.87%	85.55%	86.10%	72.49%	66.02%

testing process. In order to emulate a cross-validation process, we shuffled 10 times the sensors belonging to each clusters, so that 10 different divisions were carried out to train the network in 10 different ways.

VI. RESULTS AND ANALYSIS

In this section the results obtained after applying supervised ML algorithms to all datasets are presented. As we mentioned beforehand, new datasets had been made after obtaining the cluster predicted for each sensor, adding as an extra feature the cluster each sensor belongs to. This feature was used as the output of the supervised algorithms.

As mentioned in section 5.2.1 for both datasets from Melbourne the number of clusters was decided to fix at 6 clusters, following the elbow method after applying dimensionality reduction techniques.

To decide which of the dimensionality reduction techniques between UMAP, PCA and tSNE, fits best with our data, we followed in parallel three clustering processes for both datasets after applying these techniques. As it could be seen in Table 5 the best results were achieved after applying PCA as the dimensionality reduction method for both clusterings, so we followed the rest of the work applying this technique.

In the same way, the clustering made attending the mean activations of the sensors in different time intervals showed that a sensor was clearly different attending the first two principal components, as it is shown in Fig. 10. Thus, after analysing different confusion matrices we saw that it was classified in a random cluster, so we decided to remove it from the dataset. Moreover, after examining the activations of the sensor belonging to cluster 1 and 3 we decided to merge them, assuming that they had a closely similar activity throughout different days. After applying those changes we observed an improvement of the performance metrics of a significance of 5-6% on average.

For the clustering based on buildings' typology we saw in the confusion matrix calculated for each classification algorithm that the sensors of cluster 6 were almost always classified as part of the cluster 4, so we decided to merge those two clusters. After this change had been made, the performance metrics outperformed the former ones by a significance of 1-1.5% on average.

In Table 6 different scores for different algorithms applied to the Buildings' typology dataset based clustering are shown.

TABLE 6. Metrics for building typology dataset based clustering.

Classifier	Accuracy	Precision	Recall	F-Score	Confidence interval
SVM	93.39%	92.99%	93.24%	92.93%	92.64-94.15%
RF	94.14%	93.42%	93.92%	93.08%	93.42-95.07%
DT	88.86%	89.44%	89.05%	88.90%	87.06-90.21%
MLP	81.86%	82.50%	81.80%	81.45%	79.48-84.23%
GNB	91.96%	92.52%	91.89%	91.39%	90.74-93.19%
Adaboost	95.15%	94.28%	93.55%	93.79%	94.41-95.90%
Bagging	91.54%	90.49%	90.59%	90.17%	90.51-92.56%

TABLE 7. Metrics for time-interval mean activation dataset based clustering.

Classifier	Accuracy	Precision	Recall	F-Score	Confidence interval
SVM	87.14%	88.03%	87.32%	87.02%	85.45-88.84%
RF	85.07%	85.03%	84.93%	84.86%	83.66-86.48%
DT	87.43%	87.70%	87.32%	87.02%	85.74-89.12%
MLP	66.71%	66.92%	66.71%	66.31%	64.66-78.66%
GNB	85.71%	88.02%	85.92%	86.40%	83.93-87.50%
Adaboost	81.85%	82.09%	81.78%	81.76%	80.10-83.60%
Bagging	86.10%	86.98%	86.24%	85.83%	84.48-87.71%

The scorings are the mean values of 10-fold cross validations for 30 different seeds.

In the same way, Table 7 shows the scores for different algorithms applied to the time-interval mean activation dataset based clustering. The proceeding of obtaining the scores was identical to the previous case.

For the former dataset Adaboost algorithm with RF as base classifier outperformed the rest of the classifiers' performance metrics. In case of the latter dataset, DT was the algorithm that solved the classification issue more adequately. For both we firstly supposed the H0 hypothesis, that says that each of the mean scores obtained previously came from a normal distribution. After applying Kolmogorov-Smirnov test to all the scores obtained in every cross-validation processes (n>50) we concluded that H0 hypothesis was negligible, due to the fact that p-values obtained in both cases (7.66e-188 for buildings dataset and 1.78e-131 for Mean activations dataset) were lower than 0.05. Thus, as H1 hypothesis confirms the means come from a non normal distribution.

Consequently in order to know the statistical significance of the difference of the accuracies' means, we applied the Kruskal-Wallis test. We got the statistics of 8.08504 and 0.85582 and p-values of 0.99996 and 1.0 for the Buildings and Mean activations datasets respectively. Thus,

TABLE 8. Metrics for time-interval mean activation dataset (Madrid & Melbourne) based clustering.

Classifier	Accuracy	Precision	Recall	F-Score	Confidence interval
SVM	88.89%	89.25%	88.89%	88.70%	87.62-90.15%
RF	88.30%	88.49%	88.30%	88.31%	87.32-89.27%
DT	77.37%	77.91%	77.37%	77.36%	76.28-78.47%
MLP	64.00%	64.53%	64.00%	64.03%	62.23-65.77%
GNB	83.33%	86.04%	83.33%	84.00%	82.17-84.50%
Adaboost	88.37%	88.52%	88.37%	88.37%	87.43-89.31%
Bagging	85.93%	87.22%	85.93%	85.48%	84.31-87.54%

we can not conclude that all the means arise from different distributions.

In order to figure out the relationship between both clusterings, we analyzed the general trend that followed the sensors of each cluster in the other dataset. We concluded that the sensors belonging to the cluster weekday and weekday++ were mainly related to the Office buildings, the sensors from weekend++ to Residential apartment-House/Townhouse type of buildings and finally the ones from weekday-weekend++ to the Retail type of buildings. Obviously, this makes sense taking into account that pedestrians usually spend most of their time at work during the week and at home at the weekends. At the same time, retails are visited throughout all the week.

Finally, seeking out to emulate the performance of the clustering based on the buildings' typology by the clustering based on the sensors' activity, we enlarged the former dataset by applying the data from the city of Madrid. Table 8 shows the scores for different algorithms applied to the time-interval mean activation based dataset (Madrid & Melbourne) clustering. The proceeding of obtaining the scores was identical to the previous cases.

The best classifier in terms of accuracy, precision, recall and F-score was SVM in this case. We firstly supposed the H0 hypothesis, that says that each of the mean scores obtained previously came from a normal distribution. After applying Kolmogorov-Smirnov test to all the scores obtained in every cross-validation processes for the mentioned classifier (n>50) we concluded that H0 hypothesis was negligible, due to the fact that p-value obtained (1.32e-173) was lower than 0.05. Thus, as H1 hypothesis confirms the means come from a non normal distribution.

Consequently in order to know the statistical significance of the difference of the accuracies' means, we applied the Kruskal-Wallis test. We got the statistic of 0.0 and p-value of 1.0. Thus, we can not conclude that all the means arise from different distributions.

As the results obtained for the clustering process depending on the activity of sensors did not outperform the ones

TABLE 9. Results of accuracy of the GCN.

Results of GCN test	
Accuracy	87.00%
Confidence interval	83.416-90.584%

TABLE 10. Computation times for 10-fold Cross Validation for different algorithms.

Algorithm	Building criteria	Activity criteria
k-Means+SVM	1.41 s	4.8 s
k-Means+RF	5.14 s	115.5 s
k-Means+DT	3.9 s	6.3 s
k-Means+MLP	3.48 s	284.1 s
k-Means+GNB	1.35 s	6 s
k-Means+Adaboost	4.74 s	134.1 s
k-Means+Bagging	36.24 s	1333.8 s
GCN		709.1 s

TABLE 11. Comparative of results with the literature.

Method	Average Accuracy
Top-1 Clustering(Activity)	88.89%
Top-1 Clustering(Building Typology)	95.15%
Top-1 GCN(Activity)	87.00 %
L. Zhao et al.[5]	85.66%

based on the building typology, even if we merge data from Madrid and Melbourne, we opted for designing a graph convolutional network with the ground-truth data obtained from the clustering based on the activity patterns of sensors from Melbourne. After training for 10 different seeds each 10 divisions of the sensors mentioned in section 5.4, we obtained the following accuracies in the testing process that are summarized in Table 11.

We firstly supposed the H0 hypothesis, that says that each of the mean accuracies obtained previously came from a normal distribution. After applying Kolmogorov-Smirnov test to all the accuracies obtained in every cross-validation processes ($n > 50$) we concluded that H0 hypothesis was negligible, due to the fact that p-value obtained ($6.26e-49$) was lower than 0.05. Thus, as H1 hypothesis confirms the means come from a non normal distribution. Consequently in order to know the statistical significance of the difference of the accuracies' means, we applied the Kruskal-Wallis test. We got the p-value 0.81437. Thus, we can not conclude that all means arise from different distributions.

Attending the time needed by each method to obtain groups of buildings based in both criteria, we can observe that the application of k-Means clustering algorithm and posterior validation using supervised ML algorithms is much faster than obtaining them using a graph convolutional network. The time needed by each method for 10-fold cross validation is shown in Table 10. The environment in which all development of our work had been processed is a x64 Windows 10 Operating System equipped with a Intel Core i5-10210U working at 1,6 GHz (4,2 GHz Turbo frequency, 4 core and 8 subprocesses) and 8 GB DDR-4 RAM.

TABLE 12. Abbreviations used in this paper.

Abbreviation	Whole Phrase/Word
Acc	Accuracy
ADST-Net	Attention-based Deep Spatio-Temporal Network
AI	Artificial Intelligence
BIRCH	Balanced Iterative Recucing and Clustering Using Hierarchies
CIB	International Council for Research and Innovation in Building and Construction
CLARA	Clustering Large Applications
CNN	Convolutional Neural Network
CURE	Clustering Using Representatives
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCRNN	Diffusion Convolutional Recurrent Neural Network
DDR	Doble Data Rate Synchronous
DENCLUE	Density-based Clustering
DT	Decision Tree
F	F-Score
FC	Fully Connected
FN	False Negative
FP	False Positive
GIS	Graphic Information System
GC-LSTM	Graph Convolutional networks and Long Short-Term Memory networks
GCN	Graph Convolutional Networks
GNB	Gaussian Naive-Bayes
GNN	Graph Neural Network
IoT	Internet of Things
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Mult-Layer Perceptron
MVGCN	Multi-View Graph Convolutional Network
OD-TGAT	Origin-Destination based Temporal Graph Attention Network
PAM	Partitioning Around Medoids
PC	Principal Component
PCA	Principal Component Analysis
Pr	Precision
RAM	Random Access Memory
Re	Recall
RF	Ranndom Forest
ROCKS	Robust Clustering using Links
SAHN	Sequential Agglomerative Hierarchical Non-overlapping
STGCN	Spatio-Temporal Graph Convolutional Networks
STING	Statistical Information Grid
SVM	Supported Vector Machine
TN	True Negative
TP	True Positive
tSNE	t-Distributed Stochastic Neighbor Embedding
UMAP	Unifold Manifold Approximation and Projection
WBC	World Building Congress

After all, we obtained top accuracies of 88.89% and 95.15% in the validation of the clustering methods of sensor classification attending their main activity and the building typology surrounding them, respectively. Finally, we achieved 87.00% accuracy using GCN. All of our attempts, outperformed the results obtained in [5], where they achieved 85.66% of overall accuracy, as it is shown in Table 11.

VII. CONCLUSION AND FUTURE WORK

This paper shows the potential of clustering methods, specifically K-means algorithm, at identifying groups of sensors following different criteria. First, the typology of the buildings that were within a relatively short distance (<300 m) from each sensor was identified and counted the number of buildings that were within such distance for each type. At the same time, the activation pattern of different sensors was evaluated by obtaining the mean activation in 2-hourly time intervals throughout different days of the week.

As it has been presented, the typology of the buildings was the main factor when dividing sensors into different groups. Thus, we can deduce that those sensors were placed in different types of buildings, and so were made the clusters.

On the other hand, the variations of occupants around the sensors should be also a good feature to determine the sensor's typology. After all, the classifications made by different supervised ML algorithms show that this clustering was not as conclusive as the previous one, even if we enlarged the number of sensors collecting data by adding data from the city of Madrid. Trying to improve the classification results based on the pedestrians' activity, a graph convolutional network was performed but there was no significance in improvement of metrics.

However, it was feasible to link clusters based on building typology with the ones based on the pedestrians' activity, revealing that there is a close connection between the activity pattern of the sensors and the type of environment they are located. By this way, it would be possible to tackle different security and energy efficiency tasks by knowing only the building types of an urban zone, not needing any further information. Furthermore, institutions could alleviate a great amount of effort needed to ensure safe and energy efficient urban areas.

Following this research line next step would be forecasting spatiotemporal changes in pedestrians' patterns. By doing so, real time predictions could be made, avoiding different issues and providing cities with a higher security and energy efficiency, among other benefits.

ACKNOWLEDGMENT

The authors would like to thank the TECNALIA Member of Basque Research and Tecnology Alliance for technical support.

REFERENCES

- [1] A. Sinaeepourfard, J. Garcia, X. Masip, E. Marin, J. Cirera, G. Grau, and F. Casaus, "Estimating smart city sensors data generation: Current and future data in the city of Barcelona," in *Proc. Medit. Ad Hoc Netw. Workshop (Med-Hoc-Net)*, 15th IFIP MEDHOCNET, Vilanova i la Geltrú, Spain, Jun. 2016.
- [2] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 920–939, Mar. 2011.
- [3] L. Klein, D. Gibson, and M. Mills, *Traffic Detector Handbook*, vol. 1, no. FHWA-HRT-06-108, 3rd ed. McLean, VA, USA: Federal Highway Administration, 2006.
- [4] S. Himmel, M. Ziefle, and K. Arning, *From Living Space to Urban Quarter: Acceptance of ICT Monitoring Solutions in an Ageing Society*. Berlin, Germany: Springer, 2013.
- [5] L. Zhuo, Q. Shi, C. Zhang, Q. Li, and H. Tao, "Identifying building functions from the spatiotemporal population density and the interactions of people among buildings," *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 6, p. 247, May 2019.
- [6] J. Yoon and S. Lee, "Spatio-temporal patterns in pedestrian crashes and their determining factors: Application of a space-time cube analysis model," *Accident Anal. Prevention*, vol. 161, Oct. 2021, Art. no. 106291.
- [7] L. Hu, X. Wu, J. Huang, Y. Peng, and W. Liu, "Investigation of clusters and injuries in pedestrian crashes using GIS in Changsha, China," *Saf. Sci.*, vol. 127, Jul. 2020, Art. no. 104710.
- [8] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," Jul. 2017, *arXiv:1707.01926*.
- [9] L. M. Pfister, R. G. Thompson, and L. Zhang, "Spatiotemporal exploration of Melbourne pedestrian demand," *J. Transp. Geogr.*, vol. 95, Jul. 2021, Art. no. 103151.
- [10] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017, *arXiv:1709.05584*.
- [11] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," 2017, *arXiv:1709.04875*.
- [12] Y. Xu and D. Li, "Incorporating graph attention and recurrent architectures for city-wide taxi demand prediction," *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 9, p. 414, 2019.
- [13] Y. Qi, Q. Li, H. Karimian, and D. Liu, "A hybrid model for spatiotemporal forecasting of PM_{2.5} based on graph convolutional neural network and long short-term memory," *Sci. Total Environ.*, vol. 664, pp. 1–10, May 2019.
- [14] J. Sun, J. Zhang, Q. Li, X. Yi, Y. Liang, and Y. Zheng, "Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks," vol. 14, no. 8, 2019, pp. 1–12, *arXiv:1903.07789*.
- [15] H. Jia, H. Luo, H. Wang, F. Zhao, Q. Ke, M. Wu, and Y. Zhao, "ADST: Forecasting metro flow using attention-based deep spatial-temporal networks with multi-task learning," *Sensors*, vol. 20, no. 16, p. 4574, Aug. 2020.
- [16] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2020.
- [17] U. Saralegui, "Occupancy estimation and people flow prediction in smart environments," Facultad de Informática, Universidad del País Vasco/Euskal Herriko Unibertsitatea, San Sebastian, Spain, 2017.
- [18] IBM Cloud Education. (Oct. 20, 2020). *Convolutional Neural Networks*. [Online]. Available: <https://www.ibm.com/cloud/learn/convolutional-neuralnetworks>
- [19] L. Joyanes, I. Zahonero, M. Fernández, and L. Sánchez, *Estructura de Datos en C++*. Libro de Problemas. New York, NY, USA: McGraw-Hill, 2007.
- [20] A. Garmendia-Orbegozo, S. Noye, U. Saralegui, M. A. Anton, and J. D. Nuñez-Gonzalez, "Building typology prediction in smart cities," in *Proc. Proc. CIB W78 Inf. Technol. Construct. 39th Conf. WBC*, 2022.



ASIER GARMENDIA-ORBEGOZO (Member, IEEE) was born in Azpeitia, Gipuzkoa, Basque Country, Spain, in 1996. He received the B.S. degree in physics and electronic engineering and the M.S. degree in embedded systems engineering from the University of the Basque Country (UPV/EHU), whose Director and Co-Director are J. David Nuñez-Gonzalez and Miguel Angel Anton, respectively, where he is currently pursuing the Ph.D. degree in informatics engineering.

His research interests include the application of different data mining and machine learning techniques for gaining and generating knowledge and transferring them to edge and end-user devices.



SARAH NOYE (Member, IEEE) received the Industrial Engineering degree from the Écoles Nationales Supérieures des Mines de Nancy, France, and the Ph.D. degree in systems engineering from Imperial College London, U.K. During her Ph.D. degree, she worked with wireless sensors for building commissioning at the Center for Systems Engineering and Innovation, London, providing data analysis solutions to detect deviations between design and actual performance. In 2017, she joined

TECNALIA, where she works as a Researcher in the field of artificial intelligence algorithms. She has significant experience in various projects that apply artificial intelligence both in the design and optimization of the operation of smart buildings. She has co-directed several master's theses and student internships on artificial intelligence topics. She currently leads the artificial intelligence strategy focused on graphic systems and semantic models in TECNALIA. Since 2019, she has been co-directing a Ph.D. degree in machine learning metamodels for optimizing the operation and maintenance of buildings.



J. DAVID NUÑEZ-GONZALEZ (Senior Member, IEEE) received the Ph.D. degree in computer science, in 2016. He is an Associate Professor with the Applied Mathematics Department, University of the Basque Country. He is currently advising undergraduate, master's, and Ph.D. students. He has contributed for several publications (JCR journals, books chapters, and conference papers) and participated in two European projects (SandS from FP7 and CybSpeed from

H2020). His research interests include machine learning and artificial intelligence.

...



MIGUEL ANGEL ANTON (Senior Member, IEEE) received the B.S. degree in industrial engineering from the University of the Basque Country, Donostia-San Sebastian, Spain, in 1997, the M.S. degree in automation and industrial electronics from the University of Navarra, Donostia-San Sebastian, in 2004, and the Ph.D. degree in electronics and communications from the University of Navarra, in 2009. From 2003 to 2004, he was a Student

Researcher with the Telemedicine Group, Vicomtech Technological Centre. From 2005 to 2010, he was a Ph.D. student and a Junior Researcher in the field of bioinformatics for the development of algorithms for optimization, clustering, and pattern searching using large genomic databases with the CEIT Research Centre. Since 2011, he has been a Senior Researcher with Fundación TECNALIA Research and Innovation, Donostia-San Sebastian. His research interests include the Internet of Things (IoT), embedded intelligence in edge computing, the development of optimization algorithms, and cognitive management of buildings and infrastructures. His scientific and technological production comprises publications on bioinformatics, intelligent building management, energy management, and embedded systems.