TESIS DOCTORAL

UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental

Aplicación de Inteligencia Artificial sobre infraestructuras IoT para automatizar y optimizar los procesos de agricultura intensiva en invernaderos

Autor:
D. Juan Morales García

Directores:
Dr. D. Andrés Bueno Crespo
Dr. D. José María Cecilia Canales

Murcia, Junio de 2023

TESIS DOCTORAL

UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental

Aplicación de Inteligencia Artificial sobre infraestructuras IoT para automatizar y optimizar los procesos de agricultura intensiva en invernaderos

Autor:
D. Juan Morales García

Directores:
Dr. D. Andrés Bueno Crespo
Dr. D. José María Cecilia Canales

Murcia, Junio de 2023

TESIS POR COMPENDIO

Esta tesis doctoral es un compendio de publicaciones científicas aceptadas en revistas indexadas en el JCR. Los artículos científicos, así como sus correspondientes referencias, están listados a continuación:

- **Morales-García, J.**, Bueno-Crespo, A., Martínez-España, R., & Cecilia, J. M. (2023). Data-driven evaluation of machine learning models for climate control in operational smart greenhouses. *Journal of Ambient Intelligence and Smart Environments*, 15(1), 3-17. https://doi.org/10.3233/AIS-220441.

- **Morales-García, J.**, Bueno-Crespo, A., Martínez-España, R., Posadas, J. L., Manzoni, P., & Cecilia, J. M. (2023). Evaluation of low-power devices for smart greenhouse development. *Journal of Supercomputing*, 1-23. https://doi.org/10.1007/s11227-023-05076-8.

- **Morales-García, J.**, Bueno-Crespo, A., Martínez-España, R., García, F. J., Ros, S., Fernández-Pedauyé, J., & Cecilia, J. M. (2023). SEPARATE: A tightly coupled, seamless IoT infrastructure for deploying AI algorithms in smart agriculture environments. *Internet of Things*, 100734. https://doi.org/10.1016/j.iot.2023.100734.

- Cecilia, J. M., **Morales-García, J.**, Imbernón, B., Prades, J., Cano, J. C., & Silla, F. (2023). Using remote GPU virtualization techniques to enhance edge computing devices. *Future Generation Computer Systems*, 142, 14-24. https://doi.org/10.1016/j.future.2022.12.038.

UCAM
UNIVERSIDAD CATÓLICA DE MURCIA

EIDUCAM
Escuela Internacional de Doctorado

AUTORIZACIÓN DE LOS DIRECTORES DE LA TESIS DOCTORAL PARA SU PRESENTACIÓN Y DEFENSA

El Dr. D. Andrés Bueno Crespo y el Dr. D. José María Cecilia Canales como Directores[1] de la Tesis Doctoral titulada "Aplicación de Inteligencia Artificial sobre infraestructuras IoT para automatizar y optimizar los procesos de agricultura intensiva en invernaderos", realizada por D. Juan Morales García en la Escuela Internacional de Doctorado (EIDUCAM), autorizan su presentación a trámite en su modalidad de compendio dado que reúne las condiciones necesarias para su defensa.

Lo que firmamos, para dar cumplimiento al Real Decreto 99/2011 de 28 de enero, en Murcia, a 07 de Mayo de 2023.

CECILIA CANALES JOSE MARIA - 74002431T

Firmado digitalmente por CECILIA CANALES JOSE MARIA - 74002431T
Fecha: 2023.05.07 16:22:59 +02'00'

ANDRES |BUENO| CRESPO

Firmado digitalmente por ANDRES|BUENO|CRESPO
Nombre de reconocimiento (DN): cn=ANDRES|BUENO|CRESPO, serialNumber=33382858Z, givenName=ANDRES, sn=BUENO CRESPO, ou=CIUDADANOS, o=ACCV, c=ES
Fecha: 2023.05.07 19:23:02 +02'00'

UCAM
EIDUCAM
Escuela Internacional de Doctorado

[1]Si la Tesis está dirigida por más de un Director tienen que constar y firmar ambos.

RESUMEN

La agenda de desarrollo sostenible (*Sustainable Development Goals*, SDG) de las Naciones Unidas establece una serie de objetivos con el fin de erradicar la pobreza, proteger el planeta y asegurar la prosperidad de sus ciudadanos. Entre estos objetivos se destacan: (6) "Garantizar la disponibilidad y la gestión sostenible del agua y el saneamiento para todos", (13) "Adoptar medidas urgentes para combatir el cambio climático y sus efectos" y (15) "Proteger, restaurar y promover el uso sostenible de los ecosistemas terrestres, gestionar de forma sostenible los bosques, luchar contra la desertización, detener e invertir la degradación del suelo y frenar la pérdida de biodiversidad". Los procesos industriales y, en concreto, los procesos de agricultura intensiva, son una de las principales amenazas para cumplir con los SDGs. Sin embargo, los avances tecnológicos en materias como la Inteligencia Artificial (*Artificial Intelligence*, AI), la Computación de Alto Rendimiento (*High Performance Computing*, HPC) o el Internet de las Cosas (*Internet of Things*, IoT) permiten aumentar la productividad de estos procesos reduciendo su impacto medioambiental y ecosistémico. La investigación desarrollada en la presente tesis doctoral pretende establecer un marco de trabajo donde aprovechar los avances tecnológicos desarrollados en estas disciplinas, es decir, AI, HPC e IoT, para optimizar y reducir el impacto de los procesos industriales más agresivos para el medioambiente. En concreto, esta tesis doctoral se desarrollará en el contexto de agricultura intensiva en invernaderos, un sector de un gran valor estratégico, comercial e incluso humanitario para garantizar el acceso a los alimentos a toda la humanidad, centrándose en tres puntos clave: (1) la generación de técnicas de AI de bajo consumo que puedan ser ejecutados en plataformas con reducidas capacidades de cómputo, tales como dispositivos IoT; (2) la creación de una infraestructura que permite entrenar, desplegar y predecir con técnicas de AI que requieren de grandes capacidades de cómputo en pequeños dispositivos IoT gracias a protocolos de comunicación en tiempo real como MQTT; y (3) el aumento de las capacidades de cómputo y la eficiencia energética de los dispositivos IoT gracias a la virtualización de GPUs remotas mediante rCUDA. Los principales resultados obtenidos en relación a lo expuesto anteriormente demuestran que (1) la intersección entre la AI, HPC e IoT es todavía muy incipiente. Las cargas de cómputo del aprendizaje máquina son cada vez más altas y se diverge cada vez más de los recursos computacionales disponibles en los dispositivos de cómputo más cercanos a la captura de datos, es decir, los dispositivos de *Edge Computing*. Estas plataformas no son computacionalmente capaces de desarrollar parte de las tareas más exigentes (como, por ejemplo, el entrenamiento de técnicas de AI), limitando el éxito de su aplicación; (2) se puede crear una infraestructura auxiliar que permita desarrollar predicciones en tiempo real en dispositivos IoT, aunque el intercambio de información entre los distintos nodos de la infraestructura conlleva una latencia asumible puesto que es muy reducida; y (3) es posible ampliar las capacidades computacionales y la eficiencia energética de los

dispositivos IoT mediante el uso de técnicas de virtualización de GPUs remotas. Estas técnicas aumentan notablemente la eficiencia energética de estos dispositivos ya que se delega las operaciones de mayor carga computacional a los servidores remotos de cómputo. Si bien es cierto, el consumo total de la infraestructura aumenta notablemente a causa del gasto en comunicaciones entre los dispositivos *edge* y *cloud*. Para finalizar, destacar que la presente tesis se ha desarrollado en el proyecto retos-colaboración "Desarrollo de infraestructuras IoT de altas prestaciones contra el cambio climático basadas en inteligencia artificial" (GLOBALoT) con referencia RTC2019-007159-5, financiado por el Ministerio de Ciencia e Innovación / Agencia Estatal de Investigación, que tiene un marcado carácter tecnológico y, por tanto, se ha transferido el conocimiento obtenido, desarrollando un prototipo funcional en TRL 3-4 que ha sido desplegado en un entorno real de invernadero ofrecido por uno de los socios del proyecto, la empresa NUTRICONTROL. Los resultados obtenidos muestran un claro interés por esta tecnología, sentando las bases para automatizar y optimizar procesos mediante la Inteligencia Artificial de las Cosas (*Artificial Intelligence of Things*, AIoT) para aumentar la producción y reducir el impacto medioambiental en invernaderos inteligentes.

PALABRAS CLAVE

Inteligencia Artificial; Aprendizaje Máquina; Aprendizaje Profundo; Internet de las Cosas, Computación en el Borde; Computación de Alto Rendimiento; Invernaderos inteligentes.

ABSRTACT

The United Nations Sustainable Development Goals (SDGs) establish a series of goals aimed at eradicating poverty, protecting the planet and ensuring prosperity for its citizens. These goals include: (6) "To ensure availability and sustainable management of water and sanitation for all", (13) "To take urgent action to combat climate change and its impacts" and (15) "To protect, restore and promote sustainable use of terrestrial ecosystems, sustainably manage forests, combat desertification, halt and reverse land degradation and halt biodiversity loss". Industrial processes and, in particular, intensive agricultural processes, are one of the main threats to meeting the SDGs. However, technological advances in areas such as Artificial Intelligence (AI), High Performance Computing (HPC) or the Internet of Things (IoT) allow increasing the productivity of these processes reducing their environmental and ecosystemic impact. The research developed in this doctoral thesis aims to establish a framework where to take advantage of the technological advances developed in these disciplines, i.e. AI, HPC and IoT, to optimize and reduce the impact of the most environmentally aggressive industrial processes. Specifically, this PhD thesis will be developed in the context of intensive greenhouse agriculture, a sector of great strategic, commercial and even humanitarian value to ensure access to food for all humanity, focusing on three key points: (1) the generation of low-power AI techniques that can be executed on platforms with reduced computational capabilities, such as IoT devices; (2) the creation of an infrastructure that allows training, deploying and predicting with AI techniques that require large computational capabilities on small IoT devices thanks to real-time communication protocols such as MQTT; and (3) the increase of computational capabilities and energy efficiency of IoT devices thanks to the virtualization of remote GPUs through rCUDA. The main results obtained in relation to the above demonstrate that (1) the intersection between AI, HPC and IoT is still very nascent. The computational loads of machine learning are becoming higher and higher and increasingly diverge from the computational resources available on the computing devices closest to the data capture, i.e., edge computing devices. These platforms are not computationally capable of performing some of the most demanding tasks (such as, for example, training AI techniques), limiting the success of their application; (2) an auxiliary infrastructure can be created to develop real-time predictions in IoT devices, although the exchange of information between the different nodes of the infrastructure involves an assumable latency since it is very low; and (3) it is possible to extend the computational capabilities and energy efficiency of IoT devices through the use of remote GPU virtualization techniques. These techniques significantly increase the energy efficiency of these devices by delegating the most computationally intensive operations to remote compute servers. Although it is true, the total consumption of the infrastructure increases significantly due to the communication costs between the devices it edge and it cloud. Finally, it should be noted that this

thesis has been developed in the challenge-collaboration project “Development of high performance IoT infrastructures against climate change based on artificial intelligence” (GLOBALoT) with reference RTC2019-007159-5, funded by the Ministry of Science and Innovation / State Research Agency, which has a strong technological character and, therefore, the knowledge obtained has been transferred, developing a functional prototype in TRL 3-4 that has been deployed in a real greenhouse environment offered by one of the project partners, the company NUTRICONTROL. The results obtained show a clear interest in this technology, laying the foundations to automate and optimize processes through Artificial Intelligence of Things (AIoT) to increase production and reduce environmental impact in smart greenhouses.

KEYWORDS

Artificial Intelligence; Machine Learning; Deep Learning; Internet of Things; Edge Computing; High Performance Computing; Smart Greenhouses.

AGRADECIMIENTOS

Me gustaría agradecer a todas las personas que han hecho realidad mi sueño de realizar una tesis doctoral. Y, para ello, me gustaría dedicarles unas palabras, en especial:

A mi familia, por siempre estar apoyándome en todo lo que he necesitado y, lo más importante, por siempre estar ahí. Lo sois todo para mí, me disteis una educación, un apoyo, me soportasteis tanto en los buenos como en los malos momentos, sin pedir nunca nada a cambio. No tengo palabras para agradeceros todo lo que habéis hecho por mí, y por haberme convertido en la persona que hoy en día soy. Nunca os podré devolver todo lo que me habéis dado, me faltarían miles de años sólo para devolveros una pequeña porción de todo lo recibido. Sólo os puedo decir con el corazón en la mano: GRACIAS POR TODO.

A mi amigos, pues que a pesar de no poder haberles dedicado todo el tiempo que me hubiera gustado, incluso desapareciendo por largas temporadas, también siempre han estado ahí, apoyándome, echando unas risas cuando más lo necesitaba, y siempre confiando en que podría lograrlo.

A mi tutor y directores de tesis doctoral, gracias a los cuales he podido concluir este trabajo, gracias a su inestimable disposición, apoyo y ayuda en todo lo que he necesitado. Por haber soportado todas mis inquietudes e impaciencias, por siempre haberme guiado por el camino correcto, por todo lo que habéis hecho por mí. Recuerdo el día en que confiasteis en mí para comenzar en temas de investigación mientras estudiaba la carrera, me disteis esa oportunidad, me abristeis esa puerta que, a día de hoy, se transforma en esta tesis doctoral. Siempre os lo agradeceré.

A mis compañeros de departamento, aún recuerdo el primer día que me incorporé al departamento del Grado en Ingeniería Informática cuando llegué nervioso sin saber qué me iba a encontrar, cómo iba a ser todo a partir de ese momento, esa sensación tan extraña que sentía al ser compañero de todas aquellas personas que, en su día, fueron mis profesores y yo tenía en tan alta estima. Vosotros me recibisteis con los brazos abiertos desde el primer momento, ayudándome y apoyándome siempre en todo. Muchas gracias por todas esas risas, desayunos y comidas en la cafetería, por todos esos momentos que no tienen precio.

A mis compañeros de la UPV, puesto que, a pesar de no formar parte de su misma institución, siempre me acogieron con cariño, como si fuera uno más de ellos. He tenido la suerte de contar con toda su ayuda y experiencia, conociendo a grandes investigadores y mejores personas. Ha sido un placer poder colaborar con todos vosotros. Espero y deseo continuar con esta colaboración durante mucho tiempo.

También, agradecer la financiación de está investigación al proyecto "Desarrollo de infraestructuras IoT de altas prestaciones contra el cambio climático basadas en inte-

ligencia artificial" (GLOBALoT) con referencia RTC2019-007159-5, y a la ayuda Ramón y Cajal con referencia RYC2018-025580-IA, financiados por el ministerio de Ciencia e Innovación / Agencia Estatal de Investigación.

Por tanto, a todos y cada uno de vosotros que habéis puesto vuestro granito de arena y hecho posible este sueño: de corazón, **MUCHAS GRACIAS**.

"La alegría está en la lucha, en el esfuerzo, en el sufrimiento que supone la lucha, y no en la victoria misma".

Mahatma Gandhi (1869 - 1948).

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SIGLAS Y ABREVIATURAS

AI: Artificial Intelligence; Inteligencia Artificial.

AIoT: Artificial Intelligence of Things; Inteligencia Artificial de las Cosas.

CPU: Central Processing Unit; Unidad Central de Procesamiento.

CUDA: Compute Unified Device Architecture; Arquitectura Unificada de Dispositivos de Cómputo.

DC: Data center; Centro de Procesamiento de Datos.

DL: Deep Learning; Aprendizaje Profundo.

GPU: Graphics Processing Unit; Unidad de Procesamiento Gráfico.

HPC: High Performance Computing; Computación de Alto Rendimiento.

IoT: Internet of Things; Internet de las Cosas.

JCR: Journal Citation Reports; Informe de Citas de Revistas.

ML: Machine Learning; Aprendizaje Máquina.

MPI: Message Passing Interface; Interfaz de Paso de Mensajes.

MQTT: Message Queuing Telemetry Transport; Cola de Mensajes de Transporte Telemétrico.

NRT: Near Real Time; Casi en tiempo real.

rCUDA: Remote CUDA; CUDA Remoto.

SDG: Sustainable Development Goals; Agenda de Desarrollo Sostenible.

TPU: Tensor Processing Unit; Unidad de Procesamiento Tensorial.

TRL: Technology Readiness Levels; Niveles de Preparación Tecnológica.

CAPÍTULO I –

INTRODUCCIÓN

CAPÍTULO I – INTRODUCCIÓN

En esta sección se muestra la definición y los objetivos de esta tesis doctoral, argumentando el compendio de artículos de investigación que la componen.

1.1. Definición

Esta tesis doctoral tiene un marcado carácter multidisciplinar donde diferentes áreas de conocimiento, procedentes tanto del entorno académico como de la industria, se unen para dar respuesta a una necesidad de gran calado en la sociedad actual como son las causas y las consecuencias del cambio climático. Entre las áreas tecnológicas destacan el análisis de grandes volúmenes de datos mediante técnicas de Inteligencia Artificial (AI), el Internet de las Cosas (IoT) y la Computación de Alto Rendimiento (HPC). Aunque, sin lugar a dudas, es la unión de todas estas áreas lo que justifica el carácter innovador de la investigación e innovación desarrolladas en esta tesis doctoral, cada una de ellas en sí misma responde a una necesidad real y actual que tiene un carácter innovador *per se*. A continuación, se detalla el estado actual de la técnica de cada uno de los elementos innovadores de la tesis doctoral.

1.1.1. Internet de las Cosas y Edge computing

El concepto de IoT está basado en la idea de conectar cualquier "cosa" (dispositivo, persona, vehículo, etc.) a la red para poder recoger información de un contexto particular y poder tomar decisiones a partir del análisis de esa información [1]. El IoT es una red de redes que incluye tanto a los dispositivos interconectados a través de Internet como a las personas, los procesos y los datos, con el objetivo de transformar la información estructurada y no estructurada en experiencias de usuario más ricas, valores de negocio tangibles y comportamientos autónomos basados en análisis en tiempo real. Dos factores clave que sustentan la revolución del IoT son (1) los datos, que pueden llevar patrones ocultos, correlaciones, así como otro tipo de información valiosa, y (2) el análisis en tiempo real, ya que el conocimiento a menudo es sensible a un contexto temporal y solamente útil en un marco de tiempo específico [2]. Por tanto, el análisis en tiempo y forma de los datos provenientes de las infraestructuras de IoT es crucial si queremos transformar de manera útil este diluvio de datos en conocimiento que pueda aportar un valor añadido al dominio concreto de aplicación.

La computación en la nube (*Cloud Computing* [3]) se había establecido como la infraestructura estándar a utilizar en el campo IoT para el procesamiento de dispositivos móviles y/o de baja capacidad computacional [4]. Sin embargo, en este tipo de computación, la infraestructura física que genera los datos puede estar a una gran distancia.

En [5] se muestra que el tiempo medio de ida y vuelta desde 260 puntos repartidos geográficamente hasta sus instancias Amazon EC2 óptimas es de 74 ms. Además, a este tiempo se debe agregar la latencia del primer salto inalámbrico. Estas latencias de acceso a la nube limitan el desarrollo de aplicaciones críticas o con requerimientos de tiempo real. Por ello, en los últimos años se ha propuesto nuevamente un movimiento hacia la dispersión del cómputo (en contraposición a la concentración de cómputo requerido por la computación en la nube). De esta manera, se ha propuesto mover parte del cómputo al borde de la red (*Edge Computing* [6]). Posteriormente se ha mejorado esta dispersión del cómputo proponiendo una arquitectura de dos niveles para buscar tiempos de respuesta que mejoren la interactividad en aplicaciones móviles [7]. El primer nivel es la arquitectura en la nube actual, y un segundo nivel es una red de elementos dispersos (alojados en diferentes ubicaciones) llamados *cloudlets* que pueden ofrecer un preprocesamiento en primera instancia. La proximidad de *cloudlets* proporciona diferentes beneficios, pero de especial interés para nuestro ámbito de aplicación son dos de ellos:

1. Servicios en la nube altamente sensibles, logrando reducir la latencia de conexión extremo a extremo, proporcionando un gran ancho de banda y reduciendo la inestabilidad de los servicios ubicados en el borde. El *edge computing* brinda, a través de *cloudlets*, potencia computacional en un salto inalámbrico desde los dispositivos móviles o sensores. Este enfoque habilita el desarrollo de aplicaciones sensibles a la latencia de la red o intensivas en cómputo. Ejemplos destacados de estas aplicaciones son aplicaciones móviles interactivas o de realidad aumentada. Un ejemplo de esto es la aplicación Cloud-Vision [8] en la que se incorporan modelos predictivos de AI en dispositivos móviles con el fin de realizar un reconocimiento facial, dividiendo las tareas en diversos *cloudlets* con el fin de reducir el tiempo de ejecución. Otro ejemplo es SMARTLET [9], una aplicación móvil que distribuye las tareas de reconocimiento facial mediante técnicas de AI entre diversos *cloudlets* para reducir considerablemente el tiempo de respuesta y aumentando, por tanto, el rendimiento de la aplicación en comparación con otras arquitecturas similares propuestas.
2. Escalabilidad a través del análisis en el *edge*, reduciendo el ancho de banda requerido por las aplicaciones diseñadas para el procesamiento de información proveniente de los sensores de las redes IoT. Reducir la cantidad de información que se transfiere a la nube a través de un análisis de datos en el borde puede evitar la sobrecarga de la red y también puede ofrecer ahorros energéticos. Un ejemplo destacado de esto es el marco GigaSight [10] donde el vídeo de los dispositivos móviles solo va al *cloudlet* más cercano. El *cloudlet* ejecuta una aplicación de visión artificial y envía los resultados y algunos metadatos a la nube, lo que reduce drásticamente el ancho de banda de la aplicación.

   Ni las aplicaciones ni los sistemas IoT están preparados para este movimiento

hacia la descentralización. De hecho, los programadores desempeñan un papel fundamental en este paradigma, ya que se les necesitarán para redefinir e incluso repensar las aplicaciones móviles e IoT. En esta tesis doctoral se plantean las infraestructuras IoT como sistemas heterogéneos (con diferentes arquitecturas y componentes) y completamente distribuidos en donde las aplicaciones se deberán ejecutar teniendo en cuenta ambos factores: eficiencia y consumo energético. De hecho, el software de sistema está llamado a jugar un papel fundamental para aprovechar al máximo dichas infraestructuras, ofreciendo una versión fácil de usar de este nuevo paradigma de computación. La novedad en esta tesis doctoral se encuentra en la adaptación de técnicas de AI de tal forma que puedan ser ejecutadas sobre dispositivos IoT con limitadas capacidades de cómputo aplicadas al contexto de invernaderos inteligentes.

1.1.2. Técnicas de análisis de datos en invernaderos

En el contexto de la AI, el ML proporciona a los computadores la habilidad de aprender sin necesidad de ser programados. La idea fundamental es que se construya un programa que sea capaz de aprender una tarea determinada en base a la experiencia mediante mecanismos de ensayo/error. Por ejemplo, las técnicas de AI permiten realizar tareas tales como visión por computador [11], detección de células cancerígenas en tomografías [12], creación de modelos predictivos para anticipar cambios en las condiciones climáticas [13], detección de sequías [14], mejora en la utilización de recursos hídricos [15] u optimización de procesos agrícolas [16].

En el último lustro ha surgido una enorme proliferación en el empleo de técnicas de AI para resolver problemas de ciencia e ingeniería. Esto ha sido motivado, principalmente, gracias a la ingente cantidad de datos que los usuarios de Internet generan cada minuto (*Big Data*), a la extensión del IoT, a los nuevos frameworks (Theano [17], Caffe [18], Torch [19], MXNet [20], TensorFlow [21], etc.) que posibilitan un despliegue más rápido y eficiente en el uso de técnicas de AI, como el Machine Learning (ML) y Deep Learning (DL), y, especialmente, a los grandes avances de la tecnología de cómputo que permite procesar modelos predictivos cada vez más complejos. Estas tecnologías de cómputo están principalmente basadas hoy en día en el uso de aceleradores tipo GPU.

Los sistemas de invernaderos ubicados en la costa mediterránea cuentan con ventajas climáticas tales como radiación incidente alta y temperatura suave en invierno, además de tener un alto potencial productivo [22]. Sin embargo, la escasa incorporación de tecnología y la falta de control climático activo reducen su capacidad productiva, muy alejada de su máximo potencial y sujetos a la evolución climática local [23]. Además, la lluvia limitada y la creciente competencia por los recursos hídricos requiere optimizar la eficiencia del uso del agua de riego [24]. Por otra parte, se debe prestar especial atención al control de fugas de nutrientes para evitar impactos medioambientales [25]. De hecho, la sostenibilidad de la agricultura intensiva en invernaderos requiere

de atención urgente y dedicada a la optimización del uso de los recursos naturales, principalmente, energía y agua [26].

Por lo tanto, se demandan nuevos sistemas de invernaderos para controlar eficientemente el clima, la fertirrigación y el manejo de plagas [27]. Sin embargo, esto no es sencillo, ya que requiere un control coordinado entre todos los factores implicados (medio ambiente, fertirrigación, etc.) dentro del invernadero, que también están directamente influenciados por componentes externos como la climatología o los factores socioeconómicos (por ejemplo, el coste de la energía, el suministro de agua, etc.) [28]. Por lo tanto, un invernadero es, sin duda, un sistema complejo e incierto que depende de un gran número de variables (multivariante), cuya modelización debe ser abordada de manera holística, incluyendo todos los factores, internos y externos, que puedan estar involucrados en el proceso [29, 30]. De hecho, los invernaderos pueden aprovechar las innovaciones tecnológicas en los ámbitos de la informática, las telecomunicaciones y la agricultura para lograr una mayor eficiencia productiva, sostenibilidad y también una producción respetuosa con el medio ambiente [31].

El IoT se está aplicando ampliamente en invernaderos [32], ya que están evolucionando hacia la supervisión continua y los sistemas automatizados, que generan una gran cantidad de datos los cuales deben de ser analizados. Actualmente, este análisis, se basa principalmente en reglas de decisión, que se formulan utilizando modelos predictivos univariantes (es decir, de un único factor medioambiental) que siguen los principios de la lógica clásica, siendo el método más común el control "proporcional-integral-derivado" [33]. La principal desventaja de este método es que se centra en un único factor medioambiental y, sobre la base de unos pocos parámetros específicos, sólo se pueden tomar decisiones específicas, ignorando el resto de los factores implicados. Se han llevado a cabo algunas investigaciones recientes en el campo del ML para el desarrollo de invernaderos inteligentes. Sin embargo, estos trabajos se han basado en la definición de técnicas de ML muy limitadas, con modelos predictivos de una o muy pocas variables de entrada, generalmente obtenidas a partir de bases de datos disponibles al público o creadas de forma artificial [34].

Existen algunos trabajos que realizan análisis con técnicas de AI como redes bayesianas [35], algoritmos genéticos [36] o redes neuronales [37] para optimizar e intentar implantar sistemas de control climático o energético dentro de los invernaderos. Sin embargo, a la escasez de estos estudios hay que sumarle la unilateralidad de los mismos, donde solamente pretenden optimizar una variable dentro del invernadero. Las técnicas de DL ofrecen la capacidad de poder inferir y analizar varias variables a la vez para realizar una salida óptima del sistema. Estas técnicas se están implantando en muchas áreas, siendo una de ellas la agricultura [38]. La detección de enfermedades en plantas [39] y monitorización de cultivos [40] son algunas de las actividades donde se están utilizando estas técnicas de DL. Los resultados obtenidos por las técnicas de DL son satisfactorios, robustos y fiables para ser introducidos dentro de los sistemas de monitorización de invernaderos. Por lo tanto, estas técnicas de DL también pueden

ser aplicadas, de forma conjunta, para analizar en tiempo real todas las variables influyentes en un invernadero y poder crear un sistema de ayuda a la decisión que permita la monitorización y automatización completa de un invernadero. El diseño de estas técnicas para crear modelos predictivos multivariantes de ayuda a la decisión es una de las características innovadoras en esta tesis doctoral.

1.1.3. Aceleración de técnicas de Inteligencia Artificial

Como se ha comentado en las secciones anteriores, las técnicas de AI, como aquellas enmarcadas en las disciplinas del ML y DL, están ofreciendo soluciones novedosas en diferentes ámbitos de la sociedad. Sin embargo, los requisitos computacionales de este tipo de modelos predictivos de AI son muy elevados, limitando su aplicación a problemas donde los requisitos temporales están muy restringidos. De hecho, muchos modelos predictivos son entrenados offline, utilizando computación de alto rendimiento (HPC) que involucran el uso de cientos o incluso miles de procesadores multi-núcleo (AMD Ryzen o Intel Xeon Phi), junto a aceleradores hardware tales como las unidades de procesamiento de gráficos (GPUs) [41] o las Field-Programmable Gate Arrays (FPGAs) [42] como plataformas de cómputo programables para acelerar alguna fase del cómputo de estos modelos predictivos [43, 44, 45].

La ejecución de estos modelos predictivos de AI offline impide desarrollar aplicaciones interactivas y/o de tiempo real. En la actualidad, se están desarrollando dispositivos de propósito específico llamados circuitos integrados específicos de aplicación (ASIC) [46] para realizar ciertas tareas asociadas de estos modelos predictivos [47]. Estas tareas se restringen, normalmente, a los procesos de inferencia, asumiendo que los modelos predictivos han sido previamente entrenados en un cluster de computación. Por ejemplo, Google ha desarrollado la unidad de procesamiento de tensores (TPU) para acelerar la fase de inferencia de redes neuronales artificiales que muestran un factor de velocidad 15x-30x en comparación con la versión homóloga de CPU y GPU [48]. Asimismo, NVIDIA ha incluido procesadores específicos para la ejecución de este tipo de cargas computacionales en sus aceleradores gráficos desde su arquitectura denominada Volta [49].

Esta evolución de las plataformas de cómputo no hace más que reafirmar la tendencia en el desarrollo de procesadores hacía los sistemas heterogéneos. Esto ha provocado un cambio de enfoque de una "computación serie centrada en el rendimiento" a una "computación paralela eficiente energéticamente", en la que se hace fundamental un codiseño hardware / software de las aplicaciones para realmente aprovechar todos los recursos de los sistemas. Sin lugar a dudas, este nuevo paradigma eleva el coste del desarrollo software [50]. La novedad en esta tesis doctoral se encuentra en dotar a las infraestructuras IoT de capacidades de cómputo de alto rendimiento, basadas en GPU, sin desatender ciertas características de las infraestructuras IoT como pueden ser las

severas limitaciones energéticas.

1.1.4. Infraestructuras de virtualización de GPUs

La transición a la computación heterogénea ha delegado la principal responsabilidad de lograr rendimiento en los programadores. Este paradigma de cómputo se enmarca dentro de las plataformas heterogéneas que combinan diferentes tipos de procesadores para mejorar la eficiencia computacional y energética de estos grandes supercomputadores [51, 52]. Desarrollar aplicaciones eficientes no solo se basa en escribir un programa para realizar una tarea específica, sino que los programadores también tienen que decidir qué partes de las aplicaciones hacen uso de los aceleradores tipo GPU y qué partes se ejecutan en procesadores de propósito general como las CPUs [53, 54]. Además, se tienen que sincronizar diferentes ejecuciones y unir resultados parciales, mantener la coherencia entre las diferentes memorias de dispositivos, etc. Como se puede ver, la distribución óptima del trabajo entre todos los dispositivos dentro de un sistema heterogéneo no es trivial.

Además de la dificultad intrínseca en el diseño de aplicaciones eficientes que hacen uso de GPUs, a nivel de sistema, las GPUs también resultan complejas de gestionar. Por una parte, estos dispositivos tienen un elevado coste de adquisición. Por otra, las GPUs consumen una gran cantidad de energía para llevar a cabo sus cómputos. En este aspecto, aunque las GPUs son dispositivos muy eficientes energéticamente mientras llevan a cabo cómputos, cuando están inactivas pueden llegar a requerir una cantidad nada desdeñable de energía [55]. Además, las GPUs son dispositivos que, en general, requieren mucho espacio para su instalación, lo que provoca que la densidad de CPUs en los centros de datos (DC) que las usan se vea reducida. Finalmente, dado que las GPUs son dispositivos que aceleran algunas de las partes de una aplicación, la utilización media de estos aceleradores suele ser baja [56].

Una manera de abordar los problemas asociados con el uso de GPUs es usar técnicas de virtualización [57]. Estás técnicas pueden proporcionar mejoras significativas, ya que permiten una mayor utilización de los recursos a base de compartir un hardware entre varios usuarios, reduciendo así la cantidad requerida de instancias de ese dispositivo concreto. La virtualización también resulta muy beneficiosa para mejorar la eficiencia energética de una instalación de cómputo [58]. Como resultado, la virtualización se está adoptando cada vez más en los DC [59]. Por otra parte, existen algunas técnicas de virtualización que, en lugar de virtualizar toda la computadora, se centran en la virtualización de recursos específicos, como pueden ser las GPUs. Existen diversos middleware, como GVirtuS [60] o rCUDA [61] que permiten el uso simultáneo y remoto de GPUs compatibles con CUDA. Para habilitar la aceleración remota basada en GPUs, estas herramientas crean aceleradores virtuales en ordenadores sin GPUs, donde se ejecutan las aplicaciones aceleradas. Estos dispositivos virtuales representan

GPUs físicas ubicadas en un computador remoto que ofrece servicios de GPU. Además, se puede hacer un uso concurrente de las GPUs físicas remotas. Por lo tanto, todos los computadores del clúster pueden acceder al conjunto completo de aceleradores CUDA presentes en el clúster al mismo tiempo. Además, una aplicación de memoria compartida que se ejecuta en un único ordenador del clúster podría acceder a todas las GPUs del clúster sin utilizar herramientas como Message Passing Interface (MPI), lo que reduce, en la práctica, la complejidad de la programación de aplicaciones [62]. Como puede observarse, la virtualización remota de GPUs ofrece muchos beneficios, siendo su principal inconveniente la necesidad de un ancho de banda elevado en la red que comunica el ordenador donde se ejecuta la aplicación y el ordenador donde está instalada la GPU, así como el requerimiento de una baja latencia.

Las infraestructuras IoT, desde la perspectiva computacional, podrían verse como un clúster heterogéneo y distribuido donde las aplicaciones deben ejecutarse con restricciones de rendimiento y consumo energético. Esto es especialmente cierto cuando se quiere dotar a la infraestructura IoT de capacidades relacionadas con AI, donde es necesaria una capacidad de cómputo que, habitualmente, no está presente en este tipo de infraestructuras [63]. Por tanto, para dotar a las infraestructuras IoT de la capacidad de cómputo necesaria al tiempo que se salvaguardan los requerimientos de rendimiento y consumo, se puede hacer uso de las técnicas de virtualización remota de GPUs descritas anteriormente. No obstante, las características de cómputo y de comunicaciones presentes en las infraestructuras IoT hacen que la implantación de la virtualización remota de GPUs en ellas sea todo un reto. Precisamente, esta es una de las características innovadoras de esta tesis doctoral.

1.1.5. Resumen de los elementos innovadores de la tesis doctoral

En las secciones anteriores se han descrito los diferentes elementos tecnológicos que se integran en esta tesis doctoral.

A este respecto, se ha descrito el IoT y cómo se integra con el paradigma de *edge computing*. El elemento innovador de esta propuesta reside, principalmente, en la adaptación de modelos predictivos de AI para que puedan ser ejecutados en dispositivos cuyas capacidades de cómputo son limitadas, tales como dispositivos IoT. Además, la tecnología creada se aplicará en el escenario de agricultura de precisión en invernaderos inteligentes.

También se ha descrito el estado del arte del control climático en invernaderos, comparándolo con nuestra propuesta basada en el uso de AI. Este es uno de los grandes elementos innovadores de esta tesis doctoral, pues se contemplará el invernadero holísticamente, como un sistema complejo en el que influyen multitud de variables en lugar de una (o unas pocas) variables.

Por otra parte, se ha discutido que el uso de modelos predictivos de AI requiere

de grandes recursos computacionales; es decir, para que las infraestructuras IoT consideradas en esta tesis doctoral puedan ejecutar modelos predictivos, hay que dotarlas de la potencia computacional necesaria. Nuestra propuesta pretende dotar de esas capacidades computacionales a las infraestructuras IoT mediante el uso de la virtualización remota de aceleradores gráficos.

1.2. OBJETIVOS

El objetivo de esta tesis doctoral es desarrollar una tecnología de carácter transversal, basada en una infraestructura hardware y software, orientada a la lucha contra el cambio climático. Esta tecnología permitirá el análisis eficiente de grandes cantidades de datos en entornos IoT. Para conseguir este análisis eficiente se utilizarán algoritmos de AI combinados con mecanismos que dotarán a los entornos IoT de capacidades de cómputo de alto rendimiento típicas de grandes centros de datos. La tecnología desarrollada en esta tesis doctoral se enfocará, durante la ejecución de la misma, a la optimización de los sistemas de producción agrícola en entornos de agricultura intensiva en invernaderos inteligentes, con el objetivo de hacer un uso sostenible de los recursos naturales. Por tanto, el objetivo general es de esta tesis doctoral es *la automatización y optimización de los procesos de agricultura intensiva en invernaderos mediante el uso de Inteligencia Artificial sobre infraestructuras IoT* (OG).

Este objetivo general servirá para demostrar que el uso de nuestra tecnología puede reducir drásticamente el impacto en el medio ambiente de las acciones del ser humano, mejorando el uso de los recursos naturales, reduciendo las emisiones de CO_2 y aminorando, por tanto, las principales causas del cambio climático. Para ello, se llevará a cabo un análisis en tiempo real, y sensible al contexto, del conjunto de datos proporcionado por la infraestructura IoT.

1.2.1. Objetivos científicos

Los puntos de interés particulares se detallan a continuación en forma de objetivos científicos.

El primer objetivo científico se centra en *la optimización de sistemas de computación y comunicaciones en infraestructuras IoT estáticas* (OC1). Este objetivo científico abarca la investigación necesaria para dotar a infraestructuras IoT de las capacidades de cómputo y de comunicaciones para llevar a cabo el análisis en tiempo real de los datos proporcionados por la propia infraestructura IoT. Este análisis se llevará a cabo mediante la utilización de algoritmos de AI. La ejecución de estos algoritmos requiere de una gran potencia computacional, la cual es proporcionada en la actualidad por aceleradores de cómputo como las GPU. Para dotar a las infraestructuras IoT de esta capacidad de cómputo se propone hacer uso de técnicas de virtualización remota de GPUs, las cuales

dan acceso a GPUs potentes, instaladas fuera de la infraestructura IoT y, por tanto, sin las limitaciones, en cuanto a la energía consumida se refiere, existentes en un entorno IoT. No obstante, el acceso a dichos aceleradores remotos impone una serie de importantes condicionantes en cuanto a las comunicaciones entre la infraestructura IoT y las GPUs remotas se refiere. Resolver estas limitaciones en las comunicaciones supone un verdadero reto de investigación.

El segundo objetivo científico se centra en *la optimización de recursos naturales y energéticos en invernaderos* (OC2). Este objetivo científico abarca la investigación relacionada con el desarrollo de algoritmos de AI capaces de optimizar los procesos asociados al escenario de infraestructura IoT considerado en esta tesis doctoral. Para alcanzar este objetivo, se trabajará con datos numéricos de sensores físicos y el diseño de algoritmos de AI adecuados al ámbito de aplicación. Como se puede observar, la investigación a llevar a cabo en este objetivo científico va a ser una investigación muy centrada en actividades reales concretas. En concreto, esta investigación va a conllevar toda la dificultad asociada a la creación de modelos de sistemas reales complejos.

Estos objetivos científicos previamente mencionados están relacionados con la investigación que se llevará a cabo durante la ejecución de esta tesis doctoral.

1.2.2. Objetivos tecnológicos

La ejecución técnica de los objetivos científicos expresados anteriormente lleva consigo el siguiente objetivo tecnológico: *Desarrollo de un prototipo de invernadero autónomo para la optimización del uso de recursos naturales* (OT1). Este objetivo tecnológico está relacionado con el uso de la investigación realizada en esta tesis doctoral para crear sistemas reales que tengan un impacto en la sociedad. En concreto, este objetivo tiene como misión la integración de los algoritmos de AI fruto de la investigación desarrollada en el objetivo científico OC2 con el nuevo sistema de virtualización remota de GPUs desarrollado en el objetivo científico OC1 y, además, esta integración se llevará a cabo en el escenario IoT considerado en esta tesis doctoral. Por otra parte, dado que en este objetivo tecnológico se construirá un prototipo usando las diferentes tecnologías creadas en esta tesis doctoral, este objetivo tecnológico significará, al fin y al cabo, la demostración de que las propuestas realizadas son realmente factibles.

1.3. Fundamentación del compendio de trabajos

Es este apartado se pretende hacer un resumen de los artículos científicos presentados en esta tesis doctoral, así como la relación entre ellos y la consecución de los objetivos.

Con el objetivo de realizar un control climático en el invernadero, se realiza el

primer artículo que está relacionado con el OC1 y OC2. Consiste en el preprocesamiento de los datos recogidos de los sensores IoT desplegados en el invernadero y en el desarrollo de modelos predictivos de AI, más concretamente, de ML para realizar predicciones sobre dichos datos.

Tras analizar los resultados de dichos modelos predictivos de ML, se puede observar que, a pesar de que la calidad de la predicción obtenida es buena, está lejos de su máximo potencial al cual se podría llegar mediante la implementación de modelos predictivos de DL. Por tanto, en el segundo artículo se propone una implementación de modelos predictivos de DL, así como su adaptación para que puedan ser ejecutados en dispositivos con bajas capacidades de computo, IoT. Esta adaptación de los modelos predictivos de DL consiste en una serie de transformaciones de datos para reducir su dimensionalidad, modificaciones en la arquitectura intrínseca de los modelos predictivos estudiados (CNN y LSTM), así como su traducción a TinyML. Con esto se pretende conseguir modelos predictivos de bajo consumo que puedan ser ejecutados sobre dispositivos IoT y sirvan para automatizar y optimizar los procesos de agricultura intensiva en invernaderos.

El problema de esta adaptación de los modelos predictivos de DL es que conlleva una pérdida notable en la calidad de las predicciones y su consecuente pérdida de eficiencia en la optimización de los procesos de agricultura intensiva. Para intentar solventar esta deficiencia identificada, se desarrolla el tercer artículo, en el cual se propone *SEPARATE*: una arquitectura descentralizada, perfectamente acoplada y sin fisuras cuyo fin es desplegar algoritmos de AI en entornos IoT. Este artículo está relacionado con los objetivos OC1, OC2 y OT1 y consiste en el desarrollo de una arquitectura descentralizada que es capaz de entrenar y predecir con técnicas de AI cuyas capacidades de cómputo son elevadas en dispositivos IoT. Para ello, se propone un sistema de comunicaciones en tiempo real basado en el protocolo MQTT y en API RESTful endpoints. En concreto, cada vez que un dato es recogido por un sensor se envía a un servidor HPC que almacena el dato y re-entrena las técnicas de AI periódicamente. Simultáneamente, ese mismo dato es enviado a los dispositivos IoT encargados de automatizar y optimizar los procesos para que, una vez recibido, realicen una predicción de lo que va a suceder para así poder tomar decisiones en consecuencia. Para finalizar, existe una conexión entre el servidor HPC y los dispositivos IoT, mediante el endpoint RESTful para enviar el modelo de AI entrenado desde el servidor HPC a los dispositivos IoT.

SEPARATE, a pesar de ser una arquitectura en tiempo real, el envío de los datos y de los modelos predictivos de AI entrenados implica una latencia que, aunque es asumible puesto que es reducida, se ha intentado evitar haciendo uso de la virtualización de GPUs remotas, tal y como se puede observar en el cuarto artículo. Este artículo, relacionado con el objetivo OC1, pretende aumentar las capacidades computacionales de los dispositivos IoT mediante el uso de remote CUDA (rCUDA): una librería basada en CUDA que permite virtualizar GPUs remotas en otros dispositivos. De esta manera, se puede virtualizar una GPU de altas prestaciones en dispositivos IoT, aumentando

notoriamente sus capacidades de cómputo y reduciendo sus consumos energéticos en los propios dispositivos IoT.

La relación entre artículos-tareas-objetivos se puede observar en la figura 1.

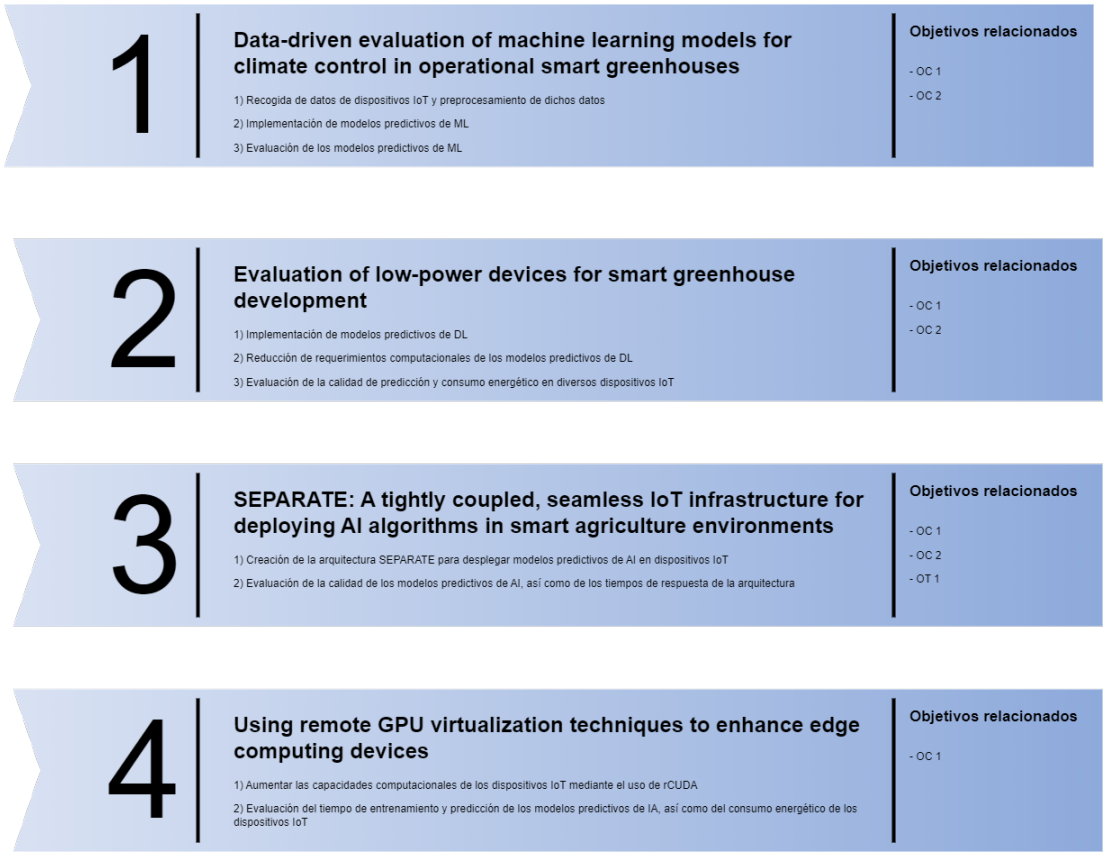


Figura 1: Relación entre los artículos científicos, las tareas desarrolladas en los mismos, y los objetivos a alcanzar.

CAPÍTULO II –

ARTÍCULOS CIENTÍFICOS

CAPÍTULO II – ARTÍCULOS CIENTÍFICOS

A continuación, se exponen los artículos científicos que componen y fundamentan esta tesis doctoral.

2.1. Data-driven evaluation of machine learning models for climate control in operational smart greenhouses

Los detalles de la primera publicación del compendio se muestran en la tabla 1.

Detalles del artículo	
Título	Data-driven evaluation of machine learning models for climate control in operational smart greenhouses
Revista	Journal of Ambient Intelligence and Smart Environments
Autores	Juan Morales-García, Andrés Bueno-Crespo, Raquel Martínez-España, José M. Cecilia
Año de publicación	2023
DOI	10.3233/AIS-220441
Estado	Publicado

Tabla 1: Detalles del primer artículo del compendio de publicaciones.

Journal of Ambient Intelligence and Smart Environments 15 (2023) 3–17
DOI 10.3233/AIS-220441
IOS Press

Data-driven evaluation of machine learning models for climate control in operational smart greenhouses

Juan Morales-García [a,*], Andrés Bueno-Crespo [a], Raquel Martínez-España [b] and José M. Cecilia [c]
[a] *Computer Science Department, Catholic University of Murcia (UCAM), ES, Spain*
E-mails: jmorales8@ucam.edu, abueno@ucam.edu
[b] *Information and Communications Engineering Department, University of Murcia (UM), ES, Spain*
E-mail: raquel.m.e@um.es
[c] *Computer and Systems Informatics Department, Polytechnic University of Valencia (UPV), ES, Spain*
E-mail: jmcecilia@disca.upv.es

Received 7 September 2022
Accepted 28 January 2023

Abstract. Nowadays, human overpopulation is stressing our ecosystems in different ways, agriculture being a critical example as different predictions point towards food shortages in the near future. Accordingly, smart farming is becoming key to the optimization of natural resources so that different crops can be grown efficiently, consuming as few resources as possible. In particular, greenhouses have proved to be an effective way of producing a high volume of vegetables/fruits in a reduced space and within a short time span. Hence, optimizing greenhouse functioning results in less water use and nutrient consumption, less energy use, faster growth, and better product quality. In this article, we carry out an in-depth analysis of different machine learning (ML) models to improve climate control in smart greenhouses. As part of the analysis of the techniques we also considered 3 ways of pre-processing the data, as well as 12-hour and 24-hour forecasting. We focus on forecasting the indoor air temperature of an operational smart greenhouse, i.e. assessing the data anomalies that are inherently present in these environments due to the instability of IoT infrastructures. Several ML models are adapted to time series forecasting to provide an overview of these techniques and to find out which one performs better in this particular scenario. Our results show that, after statistically validating the results, the Random Forest Regression technique gives the best overall result with a mean absolute error of less than 1 degree Celsius.

Keywords: Precision agriculture, artificial intelligence, machine learning, temperature forecasting, smart greenhouses

1. Introduction

Global population growth and shortages of natural products are having a transformative effect on agriculture. Precision agriculture (PA) was created as a discipline that seeks to use the latest technologies to achieve higher crop yields with less investment, greater sustainability and improved food safety [18]. PA is a general term that not only works with extensive agriculture, but also applies its novel techniques to different types of crops, including those grown in indoor facilities such as greenhouses. Along these lines, in recent years, new technologies have been introduced in greenhouses that are able to improve yields in a sustainable way [1]. A greenhouse is an agricultural

*Corresponding author. E-mail: jmorales8@ucam.edu.

structure that seeks to extend the production season by providing controlled indoor microclimatic conditions according to the type of crop [19]. Greenhouse agriculture in semi-arid regions, such as the Mediterranean region, has a high productive potential due to its climatic advantages. However, this potential can be further increased if more technological resources are integrated to control its climatic conditioning, as well as to reduce the costs involved with the use of natural resources, such as water or energy [19].

One of the most economically and environmentally relevant factors in greenhouses is climate control [36]. It is an effective tool for maintaining a satisfactory environment inside the greenhouse that meets the requirements of high-yielding crops as well as reducing energy and water consumption. Maintaining ideal conditions inside the greenhouse requires cooling and heating systems that consume between 65 and 85% of the total energy consumed in the greenhouse [20]. This is particularly relevant in semi-arid regions, it being estimated that the cooling energy consumption in the Mediterranean region is about 100,000 kWh/ha per year [40]. Therefore, one of the main objectives when designing smart greenhouses is to optimize the use of their energy resources in order to reduce their carbon footprint, and thus their environmental impact, while also increasing their economic sustainability [13].

A traditional greenhouse is defined as a predominantly metal structure for agricultural cultivation, covered with plastic sheeting. In these conventional greenhouses, they have a wired infrastructure with few resources and many communication problems, which leads to manual monitoring in many cases. These conventional practices require a lot of resources, especially human resources and possible wastage of energy due to the lack of exhaustive climate control. In addition, the lack of comprehensive climate control leads to lower crop yields, [24,39]. A few years ago, technological progress made it possible to upgrade the status of greenhouses and create smart greenhouses. These greenhouses use all the resources of technology, sensors, communication networks, framework and communication protocols, as well as data storage,analysis and processing, among other things, to obtain a better performance using a lower amount of resources, while being more economically and environmentally sustainable [22,34]. The advantages of a smart greenhouse over a conventional one include water savings, as it is possible to measure and control the exact amount of water needed by the plants. There is also a reduction in the use of pesticides, which benefits the production of organic crops, as pests can be controlled more efficiently and treated with organic products. Finally, the energy savings are significant, since actuators responsible for heating, humidifiers, shading, etc. are only activated when they are really needed and are disconnected when the optimum climate control is reached, meaning, there is a high precision in the control [3,15]. Several works can be found in the literature which deal with optimizing climate control in greenhouses. For instance Revathi, Radhakrishnan and Sivakumaran [35] described a classical control system for a greenhouse using a proportional-integral-derivative (PID) control. The main disadvantage of PID-based greenhouses is that they focus on the current state of a single environmental factor, so actions are taken a posteriori. Some ML methods have been proposed to forecast greenhouse internal variables [41,42]. However, these works are based on the definition of very limited ML techniques, using models with one or very few input variables, usually obtained from publicly available or artificially created databases that have already been curated. Indeed, current smart greenhouses rely on Internet of Things (IoT) infrastructures that are widely used for climate control [10,21,46]. These IoT deployments are based on several sensors which measure environmental conditions both inside and outside the greenhouse. However, data quality in such climatically aggressive environments is usually not good enough to develop data-based models like ML ones. In this paper, we analyze in depth the use of ML techniques used to forecast the indoor air temperature of an operational greenhouse and deal with its climate control. Several ML methods are designed for time-series regression of this variable, focusing on different data dimensions such as quality, granularity, seasonality and forecast horizon. Main contributions of the paper include the following:

- Design and optimization of up to four ML techniques for indoor temperature forecasting in an operational greenhouse. The techniques designed and adjusted for time series forecasting include the following: Autoregresive (AR), Autoregresive Integrated Moving Average (ARIMA), Random Forest Regressor (RFR) and K-Nearest Neighbors Regressor (KNNR).
- Design and tuning of data pre-processing techniques for data-curation of the raw data obtained from the sensors of an operational greenhouse in real-time.
- Different hourly granularities are considered in the forecasting, as well as a 12-hour and a 24-hour prediction.
- Comparison and analysis of the results obtained. In the analysis, different hourly granularities are considered when making the forecasts, along with forecasts at 12 and 24 hours, and also different types of data sanitization (pre-processing)

The rest of the paper is organised as follows. Section 2 shows related works under the umbrella of ML applied to forecasting climatic variables in greenhouses. Section 3 describes in detail different approaches proposed and the artificial intelligence methods used to compare and evaluate results. Section 4 shows the performance results for the pubsub solution and for the artificial intelligence models. Finally, Section 5 presents the main conclusions, and discusses future works.

2. Related works

IoT systems have revolutionized the agricultural production system in recent years. These systems allow a set of sensors to be deployed to monitor different environmental and agronomic variables so that farmers can control their crops. Particularly noteworthy is the technological proposal in greenhouses. Although greenhouses represent a great source of benefits for farmers, as they increase the yield and production of crops, they come with increased responsibility as the climatic conditions of the crops must be controlled and acted upon periodically. Since IoT systems greatly help with this monitoring and acting on the systems, research in this area is increasing [31,43].

In most applications of IoT systems applied to agriculture, monitoring and control account for most of the research, but prediction for anticipation of problems is still an understudied field [43]. Thus, in [37], the authors propose the use of low-cost irrigation controllers through software embedded in an arduino. The system takes measurements of the air and substrate to ensure optimal plant growth.In [33] a study is carried out on IoT for smart agriculture, analyzing the specific issues and challenges associated with the implementation of IoT to improve agriculture, surveying the wireless communication devices and technologies associated with IoT in agricultural and livestock applications, highlighting the difficulties associated with these solutions. In [26], a microcontroller is developed to control the microchamber of a greenhouse with the aim of increasing the productivity of two varieties of lettuce. The experimental results shows that the designed equipment worked according to the implemented programming algorithm. However, the ventilation, misting and shading actuators did not control the environmental variables, due to their undersizing, although they did control irrigation correctly. In [31], the authors present an IoT system deployed in a greenhouse to enable automated monitoring and control of temperature, humidity and light variables. The IoT system monitors the greenhouse and automatically activates actuators based on current data and pre-determined thresholds. In addition to those mentioned above, there are many applications that can be found which control and monitor greenhouses [2,30,47]. However, there are very few papers in the literature that design algorithms to forecast climatic variables inside a greenhouse. A study that uses forecasts to anticipate problems in a greenhouse is presented in [8]. The authors detail a real-time decision support system for disease prevention in a greenhouse tomato crop. This system monitors the greenhouse microclimate and then uses a system of rules to identify possible tomato diseases based on the climatic conditions to which the tomatoes have been exposed. Experimental results show that the system increases the effectiveness of climate control, while providing support for the prevention of diseases that are difficult to eradicate. In [16], an investigation to forecast the indoor temperature of a greenhouse is carried out using linear auto regressive models with external input and auto regressive moving average models with external input. Outdoor air temperature and relative humidity, global solar radiation and sky cloudiness are used as input variables for the models. The models are able to describe the greenhouse temperature evolution satisfactorily, but when the greenhouse is being ventilated, they are not accurate due to the nonlinearity of the ventilation strategies. Another work that also uses autoregressive models for temperature prediction is discussed in [48]. The problems and solutions found are similar to those already discussed. In [29], there is another work that considers autoregressive models as being ventilated to predict the indoor temperature of a greenhouse in Thailand. However, in this work, in addition to the regressive models, a neural network is considered to forecast this variable. The results indicate that the neural network in combination with autoregressive models achieves more accurate results. A similar investigation to the previous one but undertaken in Mexico is presented in [9]. In this study, the authors propose a structure of an autoregressive model together with a neural network trained by a Levenberg-Marquardt backpropagation algorithm to forecast the indoor temperature of a greenhouse. To perform this forecasting, the procedure uses external climatic variables. Another work forecasting the indoor temperature of a greenhouse is presented in [25]. To forecast the temperature it uses humidity, indoor temperature and light

intensity as input parameters. As part of the research, a BP neural network enhanced with a K-Nearest Neighbor algorithm is applied.

As can be seen, there are many studies on control and monitoring, but few very specific ones in terms of forecasting the temperature inside greenhouses. All the studies have a consensus on the importance of this variable. In the works on indoor temperature forecasting, more than one variable is used to carry out such a forecast and not all of them consider temperature data as a time series. Therefore, the design for this work uses only the temperature variable, simplifying the model as we only control and collect data on one variable, allowing the possibility of applying this system in greenhouses where technological resources are usually scarce.

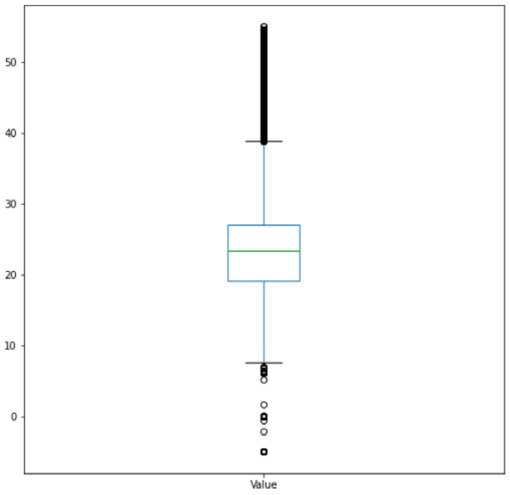
3. Materials and methods

As previously mentioned, this paper aims to design and optimize four ML techniques to forecast the indoor temperature for an operational greenhouse. This section first describes the dataset and procedures that have been used for data curation. It is important to note that we are interested in demonstrating which ML model is more tolerant to the more than likely presence of noise in the data generated by the greenhouse, so different datasets are generated. It then briefly introduces the four ML techniques used, providing details about how they have been designed and adapted to work with this time series data.

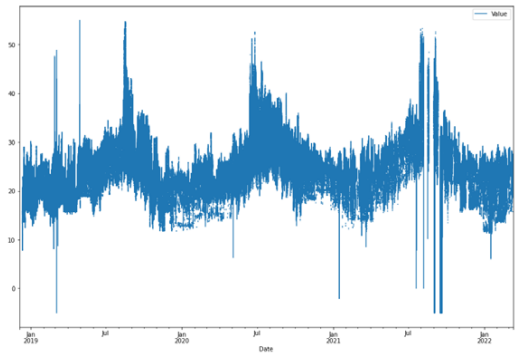
3.1. Data curation and testing dataset preparation

The greenhouse being studied has been in continuous operation since June 2018 and has generated a dataset with more than three years of real operational data for the greenhouse, including up to 21 different variables as previously mentioned. The greenhouse is operating in a semi-arid region under extreme climate conditions, particularly in summer, where temperatures of more than 45 ºC can be reached. It is also important to note that the sensor systems are energy efficient and they therefore reduce the amount of data submissions through the network by not sending data when there is no change in a particular variable. In other words, if the temperature does not change for a following 5-minute interval, the data is not published to the "Temperature" topic of the MQTT broker. Under these conditions and over such a long sampling period, a large number of errors and null values, among others, are expected to be in the dataset obtained, which may compromise the training procedures of ML models. Indeed, Fig. 1 shows a data description of the raw historical data of the greenhouse indoor temperature (TBS). Figure 1a shows the maximum value registered is 55ºC, the minimum value −5ºC, and the average value is 22.9ºC with a clear concentration between 19ºC and 28ºC (see Fig. 1b). It is highly unlikely temperatures below 0ºC or above 50ºC can occur in the area where the greenhouse is currently operating, meaning there are few out-of-range values that need to be fixed before carrying out the subsequent steps. As such, several types of action need to be taken for data sanitization to ensure a good starting point for training the ML models.

Figure 2 shows a three-day period to draw attention to the effects of applying all sanitization techniques. Figure 2a shows the raw information as retrived from the greenhouse. It is worth highlighting that there are missing values, so the first step in our sanitization procedure is to fill in the existing gaps due to the low-power mode of the sensor system. This is straightforward as it is deactivated when the temperature inside the greenhouse does not change. Consequently, the backfilling merely consists of replicating the last value collected before the gaps appeared in the time series. This procedure generates the DIRTY dataset (see Fig. 2b). The next step is to dig deeper into the raw data generated in the greenhouse by performing an outlier removal and interpolating those values that have been denoted as outliers. The identification of outliers is carried out as follows. First of all, those values that could be considered as outliers need to be identified by applying the Standard Deviation (STD) formula with a threshold of 95%. Additionally, the atypical values need to be identified, i.e. those values that cannot be considered outliers by definition, but are erroneous values within the time-series and, for this reason, a window method is used that compares the current value with the average of the last 3 values and checks that the difference does not exceed 3 degrees in absolute value. Once all the outliers or atypical values have been identified, a data imputation is carried out by performing a linear interpolation of data, i.e. the imputed value is equal to the mean of the previous value and the next value. This step ends with the generation of the CLEAN dataset (see Fig. 2c). The last dataset is obtained by



(a) Boxplot of raw data for indoor temperature data from the greenhouse, where you can see a large number of existing outliers (the black circles at the top and bottom of the box plot), the maximum and minimum values (not considered outliers) of the time series (the whiskers of the box plot), the concentration of most of the values (the box plot -quartile 1 and 3-) and the median (the inner line of the box plot -quartile 2-).

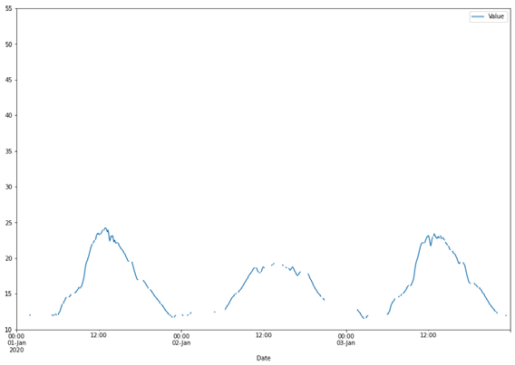


(b) Time series of raw data for indoor temperature data from the greenhouse, in which the distribution of values and their trend over time can be observed. The X-axis shows the dates on which the values were collected and the Y-axis shows the degrees Celsius for that particular date.

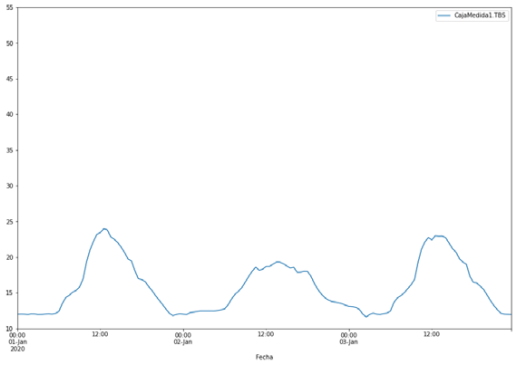
Fig. 1. Description of the raw greenhouse indoor temperature data collected by the sensors.

smoothing the CLEAN dataset. Several ML techniques benefit from data smoothing [45]. This statistical approach attempts to remove outliers from the data set, making patterns more discernible. During data compilation, data can be altered to reduce or eliminate any wide variations or other statistical noise. Smoothing helps ML models find trends or patterns that would otherwise have been missed. This approach uses simplified enhancements to better forecast various patterns. It focuses on creating a basic direction for the main data points, avoiding any volatile data and drawing a smoother curve through the data points. There are several techniques for data smoothing [12]. Among them, it is worth highlighting Kakman Filters (KF) [23] as it has been widely used for time series smoothing in many domains [4,32,44]. KF provides a sequential, unbiased, and minimum error variance estimate that works well for discrete-time filtering problems where the underlying physical phenomenon is modeled as a discrete-time process [38]. As a result, a new smooth greenhouse temperature time series was generated as shown in Fig. 2d.

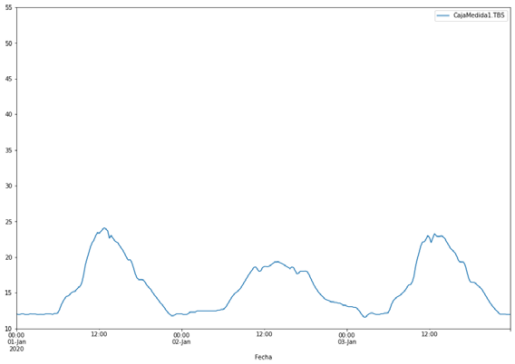
Finally, three temporal granularities are studied. Firstly, the values (i.e., 5-minute temperature samples) included in all the datasets described above are grouped into 15, 30 and 60-minute datasets. This temporal grouping is carried out by applying the arithmetic mean of the values in that interval. Secondly, the datasets are divided into summer (i.e. XXX-SUMMER) and winter (i.e. XXX-WINTER) periods. In semi-arid regions such as southern Spain, the winter periods are much more stable than the summer periods, where the thermal amplitude is much greater. It is important to note that this division is made only for the testing phase of the algorithms. The training is performed on the complete dataset. Table 1 summarizes the description of the datasets that have been used to carry out the temperature forecasting. It shows the start date of the data, the end date and the number of instances contained in each dataset. Each dataset contains the temperature values of a greenhouse between the indicated dates, distinguishing between winter and summer.



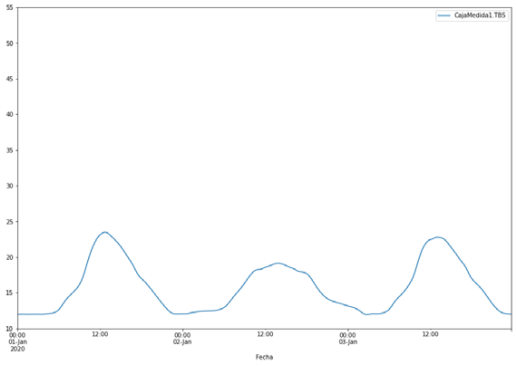
(a) Three day time series for raw data as generated by the greenhouse. At this point, no data pre-processing has been performed. The data are displayed as they are sent and collected from the greenhouse itself.



(b) Three day time series for DIRTY dataset; i.e. after filling the gaps generated by low-power mode. At this point there are no longer any empty values in the time series since they have been filled in by interpolations of data, in order to maintain the characteristics of the original time series.



(c) Three day time series for CLEAN dataset; i.e. after removal of outliers. At this point the data has been cleaned, removing outliers, atypical values, out of range values, filling missing values, etc. Accordingly, the dataset is clean, and optimized for artificial intelligence models.



(d) Three day time series for SMOOTH dataset; i.e. after smoothing. At this point the data have been smoothed by Kalman filters to eliminate possible peaks (noise) that the time series shows, meaning some imperfections that might be found in the data collection are eliminated, although it could also be the case of removing some important information from the data.

Fig. 2. Snapshot of the indoor temperature of the greenhouse, where you can see the changes in and effects of the pre-processing of the data. Since they are collected raw, the empty values are filled in (DIRTY), while the data are cleaned (CLEAN) and smoothed (SMOOTH). A window of 3 random days was chosen to show the effect of data pre-processing.

3.2. Data transformation

In order to adapt the regression models to time series models, some transformations need to be performed in the dataset, to adapt the time series to a regression problem (supervised problem) with which the model can work. This transformation consists of grouping the data in windows of a certain number of values, for which the output would be the next value to the current window. An example of this approach using windows of three values can be seen in Fig. 3. Once the data have been transformed into windows, in order to convert the time-series problem into a supervised problem, they can be used as input to the models, “X” being the set of features and “y” the target to be achieved.

Table 1
Dataset description

Datasets	Start date	End date	# Instances
CLEAN-DS-15-SUMMER	18-12-18	06-06-21	86544
CLEAN-DS-15-WINTER	18-12-18	17-01-21	73104
CLEAN-DS-30-SUMMER	18-12-18	06-06-21	43273
CLEAN-DS-30-WINTER	18-12-18	17-01-21	36553
CLEAN-DS-60-SUMMER	18-12-18	06-06-21	21637
CLEAN-DS-60-WINTER	18-12-18	17-01-21	18277
DIRTY-DS-15-SUMMER	18-12-18	06-06-21	86544
DIRTY-DS-15-WINTER	18-12-18	17-01-21	73104
DIRTY-DS-30-SUMMER	18-12-18	06-06-21	43273
DIRTY-DS-30-WINTER	18-12-18	17-01-21	36553
DIRTY-DS-60-SUMMER	18-12-18	06-06-21	21637
DIRTY-DS-60-WINTER	18-12-18	17-01-21	18277
SMOOTH-DS-15-SUMMER	18-12-18	06-06-21	86544
SMOOTH-DS-15-WINTER	18-12-18	17-01-21	73104
SMOOTH-DS-30-SUMMER	18-12-18	06-06-21	43273
SMOOTH-DS-30-WINTER	18-12-18	17-01-21	36553
SMOOTH-DS-60-SUMMER	18-12-18	06-06-21	21637
SMOOTH-DS-60-WINTER	18-12-18	17-01-21	18277

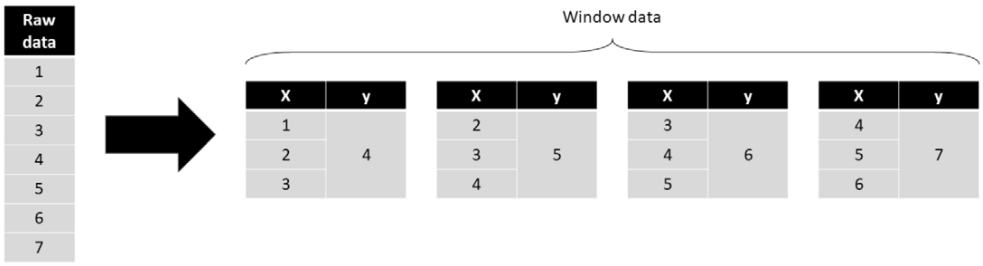


Fig. 3. Data transformation process example using a three values window.

3.3. Machine learning models

This section describes the four machine learning models used:

- **Autoregresive (AR)**:This ML model performs prediction by linear combination in a univariate model. An autoregressive model regresses on the same variable being studied, allowing it to handle a wide range of different time series patterns [27].
- **Autoregressive Integrated Moving Average (ARIMA)**: This Autoregressive model is a linear statistical model that allows regressions of statistical data to find patterns for future prediction [6]. It is a combination of autoregression (AR) which refers to the lags of the series, while moving average (MA) refers to the lags of the errors and finally integration (I) is the number of differences used to make the time series stationary.
- **Random Forest Regressor (RFR)**: Random Forest Regression (RFR) is an extension of regression decision trees (DTR), which are easy-to-interpret ML models as they generate a model tree that can be easily transformed into decision rules. This method may not be sufficient for the model to learn the characteristics of the model and in addition, DTRs have the problem that they often suffer from overfitting, so multiple decision trees created randomly are often used along with a decision system that allows the model to be improved. It is precisely this set of trees that forms the Random Forest algorithm as it can be seen as a forest of random

trees. This algorithm shows very good results in regression, and together with ANNs, it is widely used for its robustness and speed [7].

– **K-Nearest Neighbors Regressor (KNNR)**: This algorithm is based on the similarity of sample characteristics, such that a new sample is assigned a value based on its similarity to the samples in the training set. Initially, the distance between the new point and each training point is calculated. For this work, the distance has been calculated using the Euclidean distance formula, but it can be calculated by other methods, such as the Manhattan distance formula. This algorithm is rather more popular for classification problems, although it is also used in regression problems, called KNNR. A disadvantage of these methods is that the number of neighbours (k) to be considered with respect to the new sample has to be set. In our case, K = 24 has been used. The choice of the value of K is important, because if it is a very small value there may be overfitting, i.e. the classification is too close to the training set. Conversely, a very high value will make a poorly trained model [5].

4. Evaluation and discussion

In this section, an experiment is carried out, namely, the evaluation of the 5 machine learning techniques proposed in Section 3.3. For each technique, the noise tolerance is evaluated and analyzed individually, as well as the 12 and 24 hour forecasting results, taking into account the temporal granularity. Furthermore, a comparison of results using the four techniques is performed to analyze the most accurate technique for indoor temperature forecasting. The measures used for the evaluation in each of the experiments are Root Mean Square Error (`RMSE`), Mean Absolute Error (`MAE`) and coefficient of determination (R^2).

4.1. Metrics

In assessing the quality, robustness and accuracy of the models, the following metrics are used, where y_i is the real data for instance i, p_i is the forecast for instance i and N is the total number of forecasted instances:

– **Coefficient of determination ($\mathbf{R^2}$)**:This coefficient focuses on analyzing the differences between the output variable and the predictor variable. Its possible range of values is between 0 and 1, with the best models being those with a coefficient closer to 1 [28].

$$R^2 = \frac{(\sum_{i=1}^{N}(y_i - \bar{y})(p_i - \bar{p}))^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2 \sum_{i=1}^{N}(p_i - \bar{p})^2} \tag{1}$$

– **Root mean square error (RMSE)**: This value measures the amount of error between two sets of data. In other words, it compares a predicted value and an observed or known value. It is calculated as the square root of a variance. RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate a better fit [11].

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - p_i)^2}{N}} \tag{2}$$

– **Mean absolute error (MAE)**: This value is a measure of the difference between two continuous variables. Considering two sets of data (some calculated and some observed) related to the same phenomenon, the mean absolute error is used to quantify the accuracy of a prediction technique [11].

$$\text{MAE} = \frac{\sum_{i=1}^{N}|y_i - p_i|}{N} \tag{3}$$

Table 2

Results of the AR technique, values in the sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination), RMSE (root mean square error) and MAE (mean absolute error) are the metrics used in the results acquisition. RMSE and MAE are measured in degrees Celsius (°C)

Forecasting period	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.196_{0.000}$	$3.691_{0.000}$	$2.625_{0.000}$	$0.675_{0.000}$	$5.549_{0.000}$	$4.446_{0.000}$
CLEAN-DS-15-WINTER	$0.121_{0.000}$	$3.024_{0.000}$	$2.697_{0.000}$	$0.334_{0.000}$	$3.316_{0.000}$	$2.923_{0.000}$
CLEAN-DS-30-SUMMER	$0.022_{0.000}$	$3.144_{0.000}$	$2.366_{0.000}$	$0.794_{0.000}$	$4.955_{0.000}$	$4.108_{0.000}$
CLEAN-DS-30-WINTER	$0.003_{0.000}$	$3.434_{0.000}$	$3.018_{0.000}$	$0.298_{0.000}$	$3.329_{0.000}$	$2.940_{0.000}$
CLEAN-DS-60-SUMMER	$0.161_{0.000}$	$3.235_{0.000}$	$2.709_{0.000}$	$0.859_{0.000}$	$3.579_{0.000}$	$3.164_{0.000}$
CLEAN-DS-60-WINTER	$0.021_{0.000}$	$1.701_{0.000}$	$1.524_{0.000}$	$0.914_{0.000}$	$1.657_{0.000}$	$1.334_{0.000}$
DIRTY-DS-15-SUMMER	$0.193_{0.000}$	$3.625_{0.000}$	$2.621_{0.000}$	$0.674_{0.000}$	$5.392_{0.000}$	$4.323_{0.000}$
DIRTY-DS-15-WINTER	$0.121_{0.000}$	$3.110_{0.000}$	$2.773_{0.000}$	$0.335_{0.000}$	$3.342_{0.000}$	$2.940_{0.000}$
DIRTY-DS-30-SUMMER	$0.021_{0.000}$	$3.156_{0.000}$	$2.420_{0.000}$	$0.767_{0.000}$	$4.915_{0.000}$	$4.083_{0.000}$
DIRTY-DS-30-WINTER	$0.000_{0.000}$	$3.506_{0.000}$	$3.076_{0.000}$	$0.278_{0.000}$	$3.381_{0.000}$	$2.999_{0.000}$
DIRTY-DS-60-SUMMER	$0.160_{0.000}$	$3.240_{0.000}$	$2.726_{0.000}$	$0.859_{0.000}$	$3.582_{0.000}$	$3.171_{0.000}$
DIRTY-DS-60-WINTER	$0.023_{0.000}$	$1.723_{0.000}$	$1.529_{0.000}$	$0.911_{0.000}$	$1.665_{0.000}$	$1.335_{0.000}$
SMOOTH-DS-15-SUMMER	$0.149_{0.000}$	$3.615_{0.000}$	$2.603_{0.000}$	$0.733_{0.000}$	$5.518_{0.000}$	$4.448_{0.000}$
SMOOTH-DS-15-WINTER	$0.117_{0.000}$	$2.835_{0.000}$	$2.463_{0.000}$	$0.325_{0.000}$	$3.267_{0.000}$	$2.843_{0.000}$
SMOOTH-DS-30-SUMMER	$0.058_{0.000}$	$2.851_{0.000}$	$2.129_{0.000}$	$0.924_{0.000}$	$4.218_{0.000}$	$3.544_{0.000}$
SMOOTH-DS-30-WINTER	$0.000_{0.000}$	$2.737_{0.000}$	$2.313_{0.000}$	$0.624_{0.000}$	$2.647_{0.000}$	$2.225_{0.000}$
SMOOTH-DS-60-SUMMER	$0.164_{0.000}$	$3.240_{0.000}$	$2.513_{0.000}$	$0.900_{0.000}$	$4.183_{0.000}$	$3.562_{0.000}$
SMOOTH-DS-60-WINTER	$0.026_{0.000}$	$1.981_{0.000}$	$1.735_{0.000}$	$0.874_{0.000}$	$2.117_{0.000}$	$1.709_{0.000}$

4.2. Machine learning model evaluation

This section shows the results of the different ML techniques for all datasets previously described in Table 1. These results are broadly discussed in Section 4.

Table 2 shows the results obtained for all datasets using the AR model. Before going any further, it is important to highlight that the AR model is a technique that has no random components, and thus the standard deviation must be zero. The AR model obtains the worst results in terms of R^2 for all targeted ML techniques. However, the RMSE and MAE values maintain their values at a reasonable threshold; i.e. around 3°C error. The best result is obtained with the CLEAN-DS-60-WINTER and DIRTY-DS-60-WINTER datasets, reaching RMSE and MAE figures of around 1.7°C and 1.5°C respectively. It is also important to note that the 24-hour forecast obtains better metrics with respect to R^2 and maintains the values in MAE and RMSE.

Table 3 shows the results obtained for all datasets using the ARIMA model. Before going any further, it should be remembered that since it is a technique that has no random components, the standard deviation is zero like in the AR case. On the other hand, the ARIMA model obtains results that are not very stable for certain datasets. In general, it obtains more accurate results for the 24-hour forecast than for the 12-hour forecast and has a lower RMSE and MAE for CLEAN-DS-60-WINTER and DIRTY-DS-60-WINTER datasets, the error being around 1.4°C and 1.6°C.

Table 4 shows the results obtained for all datasets using the KNNR model. Before going any further, it should be noted that this is a technique using an instance-based learning model and has no random component, so the standard deviation is again zero. Regarding the results, KNNR obtains stable models, with few differences between 12-hour and 24-hour forecasting accuracy metrics. In addition, the best result in both MAE and RMSE is shown by forecasting 12-hours using the CLEAN-DS-15-WINTER dataset, obtaining RMSE and MAE values of 1.241°C and 1.021°C respectively.

Table 5 shows results obtained for all datasets using the RFR model. This technique obtains very satisfactory results with an RMSE and MAE below 0.5°C for all datasets; i.e. DIRTY, CLEAN and SMOOTH datasets. The worst-performing dataset for any preprocessing is the summer dataset with a time granularity of 60 minutes. Despite delivering worse results, we still have an RMSE and MAE of around 1°C.

Table 3

Results of the ARIMA technique, values in the sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination), RMSE (root mean square error) and MAE (mean absolute error) are the metrics used in the results acquisition. RMSE and MAE are measured in degrees Celsius (°C)

Forecasting period	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.245_{0.000}$	$11.548_{0.000}$	$8.055_{0.000}$	$0.307_{0.000}$	$27.547_{0.000}$	$22.435_{0.000}$
CLEAN-DS-15-WINTER	$0.270_{0.000}$	$9.080_{0.000}$	$7.670_{0.000}$	$0.036_{0.000}$	$38.007_{0.000}$	$27.882_{0.000}$
CLEAN-DS-30-SUMMER	$0.243_{0.000}$	$12.240_{0.000}$	$8.728_{0.000}$	$0.319_{0.000}$	$28.496_{0.000}$	$23.374_{0.000}$
CLEAN-DS-30-WINTER	$0.246_{0.000}$	$4.491_{0.000}$	$2.564_{0.000}$	$0.104_{0.000}$	$7.739_{0.000}$	$6.101_{0.000}$
CLEAN-DS-60-SUMMER	$0.103_{0.000}$	$4.430_{0.000}$	$2.623_{0.000}$	$0.559_{0.000}$	$6.068_{0.000}$	$4.939_{0.000}$
CLEAN-DS-60-WINTER	$0.011_{0.000}$	$1.470_{0.000}$	$1.322_{0.000}$	$0.939_{0.000}$	$1.429_{0.000}$	$1.169_{0.000}$
DIRTY-DS-15-SUMMER	$0.244_{0.000}$	$11.142_{0.000}$	$7.772_{0.000}$	$0.315_{0.000}$	$25.840_{0.000}$	$21.140_{0.000}$
DIRTY-DS-15-WINTER	$0.270_{0.000}$	$22.258_{0.000}$	$18.159_{0.000}$	$0.038_{0.000}$	$83.834_{0.000}$	$63.202_{0.000}$
DIRTY-DS-30-SUMMER	$0.033_{0.000}$	$4.434_{0.000}$	$3.435_{0.000}$	$0.226_{0.000}$	$6.696_{0.000}$	$5.709_{0.000}$
DIRTY-DS-30-WINTER	$0.254_{0.000}$	$5.663_{0.000}$	$3.502_{0.000}$	$0.093_{0.000}$	$10.770_{0.000}$	$8.781_{0.000}$
DIRTY-DS-60-SUMMER	$0.122_{0.000}$	$4.294_{0.000}$	$2.645_{0.000}$	$0.596_{0.000}$	$5.803_{0.000}$	$4.763_{0.000}$
DIRTY-DS-60-WINTER	$0.003_{0.000}$	$1.922_{0.000}$	$1.673_{0.000}$	$0.880_{0.000}$	$1.674_{0.000}$	$1.341_{0.000}$
SMOOTH-DS-15-SUMMER	$0.249_{0.000}$	$2.326_{0.000}$	$1.691_{0.000}$	$0.191_{0.000}$	$18.595_{0.000}$	$11.473_{0.000}$
SMOOTH-DS-15-WINTER	$0.269_{0.000}$	$4.126_{0.000}$	$2.416_{0.000}$	$0.029_{0.000}$	$8.911_{0.000}$	$7.132_{0.000}$
SMOOTH-DS-30-SUMMER	$0.238_{0.000}$	$9.814_{0.000}$	$6.527_{0.000}$	$0.347_{0.000}$	$21.423_{0.000}$	$17.535_{0.000}$
SMOOTH-DS-30-WINTER	$0.215_{0.000}$	$3.966_{0.000}$	$2.219_{0.000}$	$0.233_{0.000}$	$6.396_{0.000}$	$4.824_{0.000}$
SMOOTH-DS-60-SUMMER	$0.032_{0.000}$	$4.111_{0.000}$	$2.679_{0.000}$	$0.490_{0.000}$	$6.247_{0.000}$	$5.181_{0.000}$
SMOOTH-DS-60-WINTER	$0.040_{0.000}$	$1.828_{0.000}$	$1.580_{0.000}$	$0.901_{0.000}$	$2.055_{0.000}$	$1.640_{0.000}$

Table 4

Results of the KNNR technique, values in the sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination), RMSE (root mean square error) and MAE (mean absolute error) are the metrics used in the results acquisition. RMSE and MAE are measured in degrees Celsius (°C)

Forecasting period	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.771_{0.000}$	$3.485_{0.000}$	$2.736_{0.000}$	$0.788_{0.000}$	$3.938_{0.000}$	$3.421_{0.000}$
CLEAN-DS-15-WINTER	$0.979_{0.000}$	$1.241_{0.000}$	$1.021_{0.000}$	$0.973_{0.000}$	$1.470_{0.000}$	$1.151_{0.000}$
CLEAN-DS-30-SUMMER	$0.911_{0.000}$	$3.258_{0.000}$	$2.472_{0.000}$	$0.850_{0.000}$	$3.813_{0.000}$	$3.285_{0.000}$
CLEAN-DS-30-WINTER	$0.992_{0.000}$	$1.403_{0.000}$	$1.133_{0.000}$	$0.966_{0.000}$	$1.639_{0.000}$	$1.247_{0.000}$
CLEAN-DS-60-SUMMER	$0.887_{0.000}$	$3.211_{0.000}$	$2.470_{0.000}$	$0.860_{0.000}$	$3.748_{0.000}$	$3.234_{0.000}$
CLEAN-DS-60-WINTER	$0.964_{0.000}$	$1.347_{0.000}$	$1.019_{0.000}$	$0.954_{0.000}$	$1.662_{0.000}$	$1.201_{0.000}$
DIRTY-DS-15-SUMMER	$0.797_{0.000}$	$3.474_{0.000}$	$2.689_{0.000}$	$0.793_{0.000}$	$3.927_{0.000}$	$3.394_{0.000}$
DIRTY-DS-15-WINTER	$0.972_{0.000}$	$1.244_{0.000}$	$0.980_{0.000}$	$0.970_{0.000}$	$1.488_{0.000}$	$1.147_{0.000}$
DIRTY-DS-30-SUMMER	$0.912_{0.000}$	$3.259_{0.000}$	$2.457_{0.000}$	$0.850_{0.000}$	$3.808_{0.000}$	$3.273_{0.000}$
DIRTY-DS-30-WINTER	$0.991_{0.000}$	$1.366_{0.000}$	$1.104_{0.000}$	$0.968_{0.000}$	$1.648_{0.000}$	$1.249_{0.000}$
DIRTY-DS-60-SUMMER	$0.919_{0.000}$	$3.219_{0.000}$	$2.466_{0.000}$	$0.858_{0.000}$	$3.792_{0.000}$	$3.268_{0.000}$
DIRTY-DS-60-WINTER	$0.958_{0.000}$	$1.345_{0.000}$	$1.024_{0.000}$	$0.947_{0.000}$	$1.693_{0.000}$	$1.221_{0.000}$
SMOOTH-DS-15-SUMMER	$0.860_{0.000}$	$3.329_{0.000}$	$2.657_{0.000}$	$0.833_{0.000}$	$3.886_{0.000}$	$3.396_{0.000}$
SMOOTH-DS-15-WINTER	$0.961_{0.000}$	$1.397_{0.000}$	$1.105_{0.000}$	$0.963_{0.000}$	$1.591_{0.000}$	$1.214_{0.000}$
SMOOTH-DS-30-SUMMER	$0.935_{0.000}$	$3.108_{0.000}$	$2.493_{0.000}$	$0.879_{0.000}$	$3.734_{0.000}$	$3.284_{0.000}$
SMOOTH-DS-30-WINTER	$0.930_{0.000}$	$1.508_{0.000}$	$1.165_{0.000}$	$0.946_{0.000}$	$1.817_{0.000}$	$1.358_{0.000}$
SMOOTH-DS-60-SUMMER	$0.934_{0.000}$	$3.065_{0.000}$	$2.540_{0.000}$	$0.917_{0.000}$	$3.862_{0.000}$	$3.410_{0.000}$
SMOOTH-DS-60-WINTER	$0.839_{0.000}$	$1.959_{0.000}$	$1.755_{0.000}$	$0.905_{0.000}$	$2.267_{0.000}$	$1.821_{0.000}$

Table 5

Results of the RFR technique, values in the sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination), RMSE (root mean square error) and MAE (mean absolute error) are the metrics used in the results acquisition. RMSE and MAE are measured in degrees Celsius (°C)

Forecasting period	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.900_{0.050}$	$3.183_{0.181}$	$2.116_{0.132}$	$0.872_{0.035}$	$3.672_{0.115}$	$2.961_{0.088}$
CLEAN-DS-15-WINTER	$0.929_{0.016}$	$1.636_{0.109}$	$1.414_{0.057}$	$0.929_{0.021}$	$1.708_{0.187}$	$1.339_{0.123}$
CLEAN-DS-30-SUMMER	$0.964_{0.007}$	$2.628_{0.062}$	$1.855_{0.050}$	$0.939_{0.008}$	$3.195_{0.019}$	$2.671_{0.018}$
CLEAN-DS-30-WINTER	$0.931_{0.003}$	$1.431_{0.019}$	$1.177_{0.023}$	$0.954_{0.001}$	$1.626_{0.006}$	$1.243_{0.009}$
CLEAN-DS-60-SUMMER	$0.971_{0.012}$	$2.377_{0.247}$	$1.701_{0.115}$	$0.942_{0.016}$	$2.820_{0.135}$	$2.366_{0.084}$
CLEAN-DS-60-WINTER	$0.966_{0.003}$	$1.111_{0.051}$	$0.788_{0.024}$	$0.967_{0.003}$	$1.373_{0.039}$	$0.956_{0.021}$
DIRTY-DS-15-SUMMER	$0.860_{0.081}$	$3.273_{0.215}$	$2.173_{0.171}$	$0.857_{0.044}$	$3.719_{0.101}$	$2.998_{0.081}$
DIRTY-DS-15-WINTER	$0.954_{0.011}$	$1.661_{0.079}$	$1.399_{0.038}$	$0.928_{0.016}$	$1.772_{0.158}$	$1.368_{0.099}$
DIRTY-DS-30-SUMMER	$0.931_{0.016}$	$2.858_{0.193}$	$1.981_{0.145}$	$0.913_{0.020}$	$3.332_{0.126}$	$2.758_{0.092}$
DIRTY-DS-30-WINTER	$0.898_{0.019}$	$1.546_{0.072}$	$1.299_{0.075}$	$0.942_{0.005}$	$1.704_{0.037}$	$1.332_{0.045}$
DIRTY-DS-60-SUMMER	$0.966_{0.002}$	$2.070_{0.038}$	$1.521_{0.022}$	$0.960_{0.001}$	$2.628_{0.025}$	$2.225_{0.020}$
DIRTY-DS-60-WINTER	$0.972_{0.007}$	$0.915_{0.095}$	$0.705_{0.085}$	$0.974_{0.003}$	$1.131_{0.071}$	$0.829_{0.057}$
SMOOTH-DS-15-SUMMER	$0.899_{0.103}$	$2.437_{0.596}$	$1.711_{0.310}$	$0.924_{0.059}$	$2.976_{0.363}$	$2.454_{0.238}$
SMOOTH-DS-15-WINTER	$0.901_{0.018}$	$1.607_{0.089}$	$1.413_{0.055}$	$0.934_{0.014}$	$1.737_{0.100}$	$1.410_{0.056}$
SMOOTH-DS-30-SUMMER	$0.917_{0.024}$	$2.850_{0.175}$	$1.991_{0.091}$	$0.907_{0.020}$	$3.191_{0.195}$	$2.650_{0.124}$
SMOOTH-DS-30-WINTER	$0.781_{0.037}$	$1.691_{0.126}$	$1.355_{0.103}$	$0.879_{0.020}$	$1.623_{0.127}$	$1.267_{0.093}$
SMOOTH-DS-60-SUMMER	$0.941_{0.004}$	$2.977_{0.048}$	$2.250_{0.020}$	$0.917_{0.006}$	$3.625_{0.065}$	$3.062_{0.042}$
SMOOTH-DS-60-WINTER	$0.893_{0.009}$	$1.593_{0.031}$	$1.168_{0.019}$	$0.910_{0.006}$	$1.935_{0.108}$	$1.399_{0.082}$

Table 6

Results of the paired Friedman's statistical test (p-value row) and p-value adjustment with the Bonferroni test to find the best performing techniques

	RRF-ARIMA	RRF-AR	KNN-AR
p-value	0.000	0.000	0.000
p-value adj.	0.000	0.000	0.001

The analyses of the results were statistically validated using Friedman's test [17] for two-to-two comparisons with the results of Friedman's test before being adjusted with Dun Bonferroni's post hoc test [14]. In the statistical tests, the MAE value was used. The first statistical analysis was carried out to find out whether there are significant differences between the 12-hour and 24-hour forecasts. This analysis indicates with a confidence level of 95% that there are no significant differences between the techniques when forecasting at 12 and 24 hours.

4.3. Discussion

Analysing whether there are significant differences between the results when considering the type of pre-processing, the Bonferroni test indicates that with a 95% confidence level, there are significant differences between results achieved by ML methods with DIRTY and SMOOTH datasets and between CLEAN and SMOOTH datasets, with p-values of 0.001 and 0.0 respectively. However, there are no significant differences between the results when using DIRTY and CLEAN datasets. This suggests that our DIRTY dataset is good enough and does not need any additional preprocessing to obtain good forecasting accuracy.

Table 6 shows the results of the p-values and the p-values adjusted with the Bonferroni test to identify whether there are significant differences between the different techniques. Only p-values with significant differences are shown. The statistical tests indicate with a 95% confidence level that, for the MAE value, the best performance is for the KNNR and RFR techniques, as there are no significant differences between them. The RFR technique obtains better results, i.e. lower MAE, than the other techniques (ARIMA and AR).

Results achieved by all targeted ML techniques are satisfactory and their performance is stable and robust. However, there are some techniques that perform better in the general scenario. In particular, the worst performing techniques are the autoregressive methods; i.e. the AR and ARIMA techniques falling a long way short of the KNN and RFR, which perform much better in this case. In general terms, the MAE and RMSE error metrics are around 1–1.5°C, except for RFR where the result is a little lower, obtaining a small standard deviation and error values below 1°C. It is also important to note that most techniques do not show large differences between the 12-hour and 24-hour forecasting.

Regarding the dataset curation procedure impact on performance of ML methods, results obtained by using DIRTY and CLEAN datasets are very similar. The CLEAN dataset may offer slightly better performance in ML techniques, while The SMOOTH dataset always offers worse performance than its counterpart version (i.e., CLEAN and DIRTY) because the Kalman filter used removes relevant information that is necessary for these methods to learn and hence the results are more unstable and with a greater forecasting error.

It is also important to note that datasets evaluated in winter always perform better than those evaluated during the summer. This is due to the exponential variation and increase in summer temperatures within a short time. When temperatures are moderate, ML techniques perform very well, but when they are very high, the techniques result in a greater error. This is not a relevant problem, since in greenhouses the indoor temperatures to be forecasted are moderate and in these cases the techniques perform very satisfactorily.

Summing up, the RFR technique can be considered the best performer for forecasting temperature at both 12 and 24 hours. In addition, the SMOOTH curation procedure performs worse in all ML techniques than the other two curation procedures (DIRTY and CLEAN), whose results do not differ significantly. Taking this into account, we conclude that it is not necessary to perform preprocessing of the data to obtain good forecasting accuracy, this is to obtain low prediction error.

Finally, it is important to note that this work uses a methodology that can be applied independently of the technology or techniques used. In this methodology, the main steps consist of obtaining a data collection, initially to create a history of the data and subsequently to carry out analysis and inference. After having the information collection system, it is important to have a pre-processing method to ensure the quality of the data. Subsequently, analysis techniques and pattern extraction from the data are interesting to obtain information for decision making by ML models. Lastly, once the best prediction model and the best granularity of both temporal and long-term prediction have been analysed, such a model could then be implemented in a decision support system.

5. Conclusion and future work

Intelligent agriculture in greenhouses is currently a highly researched topic. This is due to the fact that its application encompasses different factors including cost optimization, resource optimization as well as increased sustainability. In this work, we tackled the problem of forecasting the indoor temperature in the greenhouse in advance, armed with the knowledge of the temperatures for previous time slots. This was done by applying machine learning techniques to find the most accurate models, with the lowest error, when predicting the temperature 12 and 24 hours in advance. In addition, we also performed three types of data preprocessing to analyse the best of them, always aiming to improve the quality of the data and indirectly the accuracy of the models. After data preprocessing, a total of 5 machine learning techniques were evaluated to analyse the best indoor temperature forecasting model for the greenhouse. After collecting and processing the results, a statistical analysis was carried out which concludes that preprocessing the data does not improve the results, i.e. that the forecasts at 12 and 24 hours are equally reliable and that the technique which obtains the best results is Random Forest Regressor.

As for future work, we have the possibility of including new variables to reduce the temperature forecasting error by creating multivariate models, as well as making use of the actuators that optimize the greenhouse temperature in these forecasts,.

Ethical approval

Not applicable.

Conflict of interest

The authors declare they do not have any conflict of interest.

Authors' contributions

Conceptualization, J.M.G., A.B.C., R.M.E, J.M.C.; methodology, J.M.G., A.B.C, R.M.E; software, J.M.G.; validation, J.M.C., R.M.E. and A.B.C.; formal analysis, A.B.C., J.M.C, and R.M.E; investigation, J.M.G. and R.M.E.; writing—original draft preparation, J.M.G., R.M.E, J.M.C and A.B.C.; writing—review and editing, R.M.E, A.B.C. and J.M.C.; visualization, J.M.G and A.B.C.; supervision, J.M.C. and A.B.C.; project administration, J.M.C; funding acquisition, J.M.C. and A.B.C.

All authors have read and agreed to the published version of the manuscript.

Funding

This work is derived from R&D projects RTC2019-007159-5, as well as the Ramon y Cajal Grant RYC2018-025580-I, funded by MCIN/AEI/10.13039/501100011033, "FSE invest in your future" and "ERDF A way of making Europe".

References

[1] Y. Achour, A. Ouammi and D. Zejli, Technological progresses in modern sustainable greenhouses cultivation as the path towards precision agriculture, *Renewable and Sustainable Energy Reviews* **147** (2021), 111251. doi:10.1016/j.rser.2021.111251.

[2] M.A. Akkaş and R. Sokullu, An IoT-based greenhouse monitoring system with Micaz motes, *Procedia computer science* **113** (2017), 603–608. doi:10.1016/j.procs.2017.08.300.

[3] B. Alhnaity, S. Pearson, G. Leontidis and S. Kollias, Using deep learning to predict plant growth and yield in greenhouse environments, in: *International Symposium on Advanced Technologies and Management for Innovative Greenhouses: GreenSys2019 1296*, 2019, p. 425–432.

[4] F. Avanzi, Z. Zheng, A. Coogan, R. Rice, R. Akella and M.H. Conklin, Gap-filling snow-depth time-series with Kalman filtering-smoothing and expectation maximization: Proof of concept using spatially dense wireless-sensor-network data, *Cold Regions Science and Technology* **175** (2020), 103066. doi:10.1016/j.coldregions.2020.103066.

[5] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[6] G.E. Box, G.M. Jenkins, G.C. Reinsel and G.M. Ljung, *Time Series Analysis: Forecasting and Control*, Wiley, 2015.

[7] L. Breiman and A. Cutler, Random Forests http://www.stat.berkeley.edu/users/breiman, *RandomForests/cc_home.htm (diakses tgl 27 Juli 2016)*, 2005.

[8] J. Cañadas, J.A. Sánchez-Molina, F. Rodríguez and I.M. del Águila, Improving automatic climate control with decision support techniques to minimize disease effects in greenhouse tomatoes, *Information Processing in Agriculture* **4**(1) (2017), 50–63. doi:10.1016/j.inpa.2016.12.002.

[9] A. Castañeda-Miranda and V.M. Castaño, Smart frost control in greenhouses by neural networks models, *Computers and Electronics in Agriculture* **137** (2017), 102–114. doi:10.1016/j.compag.2017.03.024.

[10] A. Castañeda-Miranda and V.M. Castaño-Meneses, Internet of things for smart farming and frost intelligent control in greenhouses, *Computers and Electronics in Agriculture* **176** (2020), 105614. doi:10.1016/j.compag.2020.105614.

[11] T. Chai and R.R. Draxler, Root mean square error (RMSE) or mean absolute error (MAE), *Geoscientific Model Development Discussions* **7**(1) (2014), 1525–1534.

[12] S.F. Chen and R. Rosenfeld, A survey of smoothing techniques for ME models, *IEEE transactions on Speech and Audio Processing* **8**(1) (2000), 37–50. doi:10.1109/89.817452.

[13] W.-H. Chen and F. You, Smart greenhouse control under harsh climate conditions based on data-driven robust model predictive control with principal component analysis and kernel density estimation, *Journal of Process Control* **107** (2021), 103–113. doi:10.1016/j.jprocont.2021.10.004.

[14] O.J. Dunn, Multiple comparisons using rank sums, *Technometrics* **6**(3) (1964), 241–252. doi:10.1080/00401706.1964.10490181.

[15] A. Escamilla-García, G.M. Soto-Zarazúa, M. Toledano-Ayala, E. Rivas-Araiza and A. Gastélum-Barrios, Applications of artificial neural networks in greenhouse technology and overview for smart agriculture development, *Applied Sciences* **10**(11) (2020), 3835. doi:10.3390/app10113835.

[16] H.U. Frausto, J. Pieters and J. Deltour, Modelling greenhouse temperature by means of auto regressive models, *Biosystems Engineering* **84**(2) (2003), 147–157. doi:10.1016/S1537-5110(02)00239-8.

[17] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, *The Annals of Mathematical Statistics* **11**(1) (1940), 86–92. doi:10.1214/aoms/1177731944.
[18] R. Gebbers and V.I. Adamchuk, Precision agriculture and food security, *Science* **327**(5967) (2010), 828–831. doi:10.1126/science.1183899.
[19] M. Ghoulem, K. El Moueddeb, E. Nehdi, R. Boukhanouf and J.K. Calautit, Greenhouse design and cooling technologies for sustainable food cultivation in hot climates: Review of current practice and future status, *Biosystems Engineering* **183** (2019), 121–150. doi:10.1016/j.biosystemseng.2019.04.016.
[20] S. Gorjian, F. Calise, K. Kant, M.S. Ahamed, B. Copertaro, G. Najafi, X. Zhang, M. Aghaei and R.R. Shamshiri, A review on opportunities for implementation of solar energy technologies in agricultural greenhouses, *Journal of Cleaner Production* **285** (2021), 124807. doi:10.1016/j.jclepro.2020.124807.
[21] M.A. Guillén-Navarro, R. Martínez-España, B. López and J.M. Cecilia, A high-performance IoT solution to reduce frost damages in stone fruits, *Concurrency and Computation: Practice and Experience* **33**(2) (2021), e5299.
[22] Y. Kaluarachchi, Potential advantages in combining smart and Green infrastructure over silo approaches for future cities, *Frontiers of Engineering Management* **8**(1) (2021), 98–108. doi:10.1007/s42524-020-0136-y.
[23] Y. Kim and H. Bang, Introduction to Kalman filter and its applications, *Introduction and Implementations of the Kalman Filter* **1** (2018), 1–16.
[24] H. Li, Y. Guo, H. Zhao, Y. Wang and D. Chow, Towards automated greenhouse: A state of the art review on greenhouse monitoring methods and technologies based on Internet of things, *Computers and Electronics in Agriculture* **191** (2021), 106558. doi:10.1016/j.compag.2021.106558.
[25] X. Li, X. Zhang, Y. Wang, Y.-F. Chen et al., *Temperature Prediction Model for Solar Greenhouse Based on Improved BP Neural Network*, Journal of Physics: Conference Series, Vol. 1639, IOP Publishing, 2020, p. 012036.
[26] A.C. Marques Filho, J.P. Rodrigues, S.D.S. de Medeiros and S.R.R. de Medeiros, Development of an electronic controller for lettuce production in greenhouses, *Revista de Agricultura Neotropical* **7**(3) (2020), 65–72. doi:10.32404/rean.v7i3.4043.
[27] T.C. Mills and T.C. Mills, *Time Series Techniques for Economists*, Cambridge University Press, 1991.
[28] N.J. Nagelkerke et al., A note on a general definition of the coefficient of determination, *Biometrika* **78**(3) (1991), 691–692. doi:10.1093/biomet/78.3.691.
[29] S. Patil, H. Tantau and V. Salokhe, Modelling of tropical greenhouse temperature by auto regressive and neural network models, *Biosystems engineering* **99**(3) (2008), 423–431. doi:10.1016/j.biosystemseng.2007.11.009.
[30] A. Pawlowski, J. Sánchez-Molina, J. Guzmán, F. Rodríguez and S. Dormido, Evaluation of event-based irrigation system control scheme for tomato crops in greenhouses, *Agricultural Water Management* **183** (2017), 16–25. doi:10.1016/j.agwat.2016.08.008.
[31] J.S. Raj and J.V. Ananthi, Automation using IoT in greenhouse environment, *Journal of Information Technology* **1**(1) (2019), 38–47.
[32] L. Ralaivola and F. d'Alché-Buc, Time series filtering, smoothing and learning using the kernel Kalman filter, in: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*, Vol. 3, IEEE, 2005, pp. 1449–1454. doi:10.1109/IJCNN.2005.1556088.
[33] P.P. Ray, Internet of things for smart agriculture: Technologies, practices and future direction, *Journal of Ambient Intelligence and Smart Environments* **9**(4) (2017), 395–420. doi:10.3233/AIS-170440.
[34] R. Rayhana, G. Xiao and Z. Liu, Internet of things empowered smart greenhouse farming, *IEEE Journal of Radio Frequency Identification* **4**(3) (2020), 195–211. doi:10.1109/JRFID.2020.2984391.
[35] S. Revathi, T.K. Radhakrishnan and N. Sivakumaran, Climate control in greenhouse using intelligent control algorithms, in: *2017 American Control Conference (ACC)*, IEEE, 2017, pp. 887–892. doi:10.23919/ACC.2017.7963065.
[36] H. Ritchie and M. Roser, *CO_2 and Greenhouse Gas Emissions, Our World in Data*, 2020, https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions.
[37] Y.A. Rivas-Sánchez, M.F. Moreno-Pérez and J. Roldán-Cañas, Environment control with low-cost microcontrollers and microprocessors: Application for Green walls, *Sustainability* **11**(3) (2019), 782. doi:10.3390/su11030782.
[38] S. Sarkka, A. Vehtari and J. Lampinen, Time series prediction by Kalman smoother with cross-validated noise density, in: *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, Vol. 2, IEEE, 2004, pp. 1653–1657. doi:10.1109/IJCNN.2004.1380209.
[39] M. Shang, G. Tian, L. Qin, J. Zhao, H. Ruan and F. Wang, Greenhouse wireless monitoring system based on the zigbee, in: *International Conference on Computer and Computing Technologies in Agriculture*, Springer, 2013, pp. 109–117.
[40] M. Soussi, M.T. Chaibi, M. Buchholz and Z. Saghrouni, *Comprehensive Review on Climate Control and Cooling Systems in Greenhouses Under Hot and Arid Conditions*, Vol. 12, Agronomy, 2022, p. 626.
[41] A.F. Subahi and K.E. Bouazza, An intelligent IoT-based system design for controlling and monitoring greenhouse temperature, *IEEE Access* **8** (2020), 125488–125500. doi:10.1109/ACCESS.2020.3007955.
[42] M. Taki, S.A. Mehdizadeh, A. Rohani, M. Rahnama and M. Rahmati-Joneidabad, Applied machine learning in greenhouse simulation; new application and analysis, *Information processing in agriculture* **5**(2) (2018), 253–268. doi:10.1016/j.inpa.2018.01.003.
[43] J.M. Talavera, L.E. Tobón, J.A. Gómez, M.A. Culman, J.M. Aranda, D.T. Parra, L.A. Quiroz, A. Hoyos and L.E. Garreta, Review of IoT applications in agro-industrial and environmental fields, *Computers and Electronics in Agriculture* **142** (2017), 283–297. doi:10.1016/j.compag.2017.09.015.
[44] S. Tavakoli, H. Fasih, J. Sadeghi and H. Torabi, Kalman filter-smoothed random walk based centralized controller for multi-input multi-output processes, *International Journal of Industrial Electronics, Control and Optimization* **2**(2) (2019), 155–166.
[45] F. Terroso-Sáenz, A. Muñoz, J. Fernández-Pedauye and J.M. Cecilia, Human mobility prediction with region-based flows and water consumption, *IEEE Access* **9** (2021), 88651–88663. doi:10.1109/ACCESS.2021.3090582.

[46] A. Tzounis, N. Katsoulas, T. Bartzanas and C. Kittas, Internet of Things in agriculture, recent advances and future challenges, *Biosystems engineering* **164** (2017), 31–48. doi:10.1016/j.biosystemseng.2017.09.007.
[47] M.A. Zamora-Izquierdo, J. Santa, J.A. Martínez, V. Martínez and A.F. Skarmeta, Smart farming IoT platform based on edge and cloud computing, *Biosystems engineering* **177** (2019), 4–17. doi:10.1016/j.biosystemseng.2018.10.014.
[48] W. Zhang, R. Zhou, C. Zhu et al., Greenhouse temperature simulation based on ARX model, *Jiangsu Journal of Agricultural Sciences* **29**(1) (2013), 46–50.

2.2. Evaluation of low-power devices for smart greenhouse development

Los detalles de la segunda publicación del compendio se muestran en la tabla 2.

Detalles del artículo	
Título	Evaluation of low-power devices for smart greenhouse development
Revista	Journal of Supercomputing
Autores	Juan Morales-García, Andrés Bueno-Crespo, Raquel Martínez-España, Juan-Luis Posadas, Pietro Manzoni, José M. Cecilia
Año de publicación	2023
DOI	10.1007/s11227-023-05076-8
Estado	Publicado

Tabla 2: Detalles del segundo artículo del compendio de publicaciones.

The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05076-8

Evaluation of low-power devices for smart greenhouse development

Juan Morales-García[1] · Andrés Bueno-Crespo[1] · Raquel Martínez-España[2] · Juan-Luis Posadas[3] · Pietro Manzoni[3] · José M. Cecilia[3]

Accepted: 20 January 2023

Abstract

The combination of Artificial Intelligence and the Internet of Things (AIoT) is enabling the next economic revolution in which data and immediacy are at the key players. Agriculture is one of the sectors that can benefit most from the use of AIoT to optimise resources and reduce its environmental footprint. However, this convergence requires computational resources that enable the execution of AI workloads, and in the context of agriculture, ensuring autonomous operation and low energy consumption. In this work, we evaluate TinyML and edge computing platforms to predict the indoor temperature of an operational greenhouse in situ. In particular, the computational/energy trade-off of these platforms is assessed to analyse whether their use in this context is feasible. Two artificial neural networks are adapted to these platforms to predict the indoor temperature of the greenhouse. Our results show that the microcontroller-based devices can offer a competitive and energy-efficient computational alternative to more traditional edge computing approaches for lightweight ML workloads.

Keywords Artificial Intelligence · Edge computing · Time series forecast · TinyML · CPU-GPU Performance · Power consumption

1 Introduction

The Internet of Things (IoT) is a major player in the digital revolution, enabling devices and humans to interact with each other in real time [1]. This interaction generates large amount of data whose analysis can provide insights from uncovering hidden patterns, correlations between variables, etc. [2]. However, knowledge is only valid as long as it is generated at the right time to enable the right decisions to

✉ Juan Morales-García
jmorales8@ucam.edu

Extended author information available on the last page of the article

Published online: 05 February 2023

be made. Therefore, enabling efficient analysis of this huge amount of data generated by the IoT is crucial to transform this deluge of data into meaningful information.

Machine learning (ML) algorithms are showing their potential for extracting knowledge from large amounts of data [3]. Traditionally, ML algorithms have been executed in supercomputers, where performance prevails over energy efficiency [4]. However, when performance is not the only concern, other approaches are feasible. For instance, edge/fog computing [5] has been approached towards decentralisation, where initial computations on data are carried out in (or close to) the data capture devices. In fact, the edge computing paradigm is providing (1) energy savings by avoiding sending and processing data in the cloud, (2) highly responsive applications and services for mobile environments, (3) highly scalable systems, thanks to the distribution of processing units, (4) guaranteed privacy policies for the IoT and (5) disconnection tolerant systems as transient connection interruptions can be masked.

IoT devices have a limited power budget at this level of the network, as they typically rely on batteries or energy harvesters, leading to ultra-low power approaches. This limited power scenario translates into a major limitation for many components of the architecture, especially energy-intensive components such as wireless transmitters or even processing capabilities [6]. A new trend, called TinyML [7], has recently emerged at the intersection of ML, IoT and computing platforms. This trend aims to leverage microcontroller units (MCUs) that are available in all devices across the IoT ecosystem, from sensor data collection and actuation, to information transfer and reception [8]. In the IoT ecosystem, MCUs had been relegated to information transfer, without taking advantage of the existing computational potential for heavier workloads, which was transferred to the edge, fog or cloud. A major factor restricting the computational use of MCUs is the limited memory size of these devices. In particular, for running ML workloads such as deep learning (DL) [9], artificial neural networks (ANNs) [10], or reinforcement learning [11] that require a certain amount of memory space.

In this paper, we evaluate different edge computing and TinyML platforms for the execution of several ANNs for time series forecasting. Performance and power evaluation is provided for up to four edge computing devices, namely Arduino microcontrollers, Raspberry PI and two Nvidia Jetson edge computing platforms. Particularly, two widely used neural network models are analysed; i.e. multi-layer perceptron (MLP) and the convolutional neural network (CNN) for the prediction of the indoor temperature of an operational greenhouse. The development of smart greenhouses is a great benchmarking to assess the intersection between edge computing and artificial intelligence. These environments are often isolated and in very aggressive meteorological conditions (e.g. high temperatures), and where internet access and power supply are not always guaranteed. Therefore, the development of smart greenhouses must be designed with these constraints in mind, guaranteeing the autonomy and continuous operability of the greenhouse. Therefore, the research question of this paper focuses on finding out if it is possible to have neural network-based time series prediction systems in fully autonomous greenhouses. Our initial hypothesis is that with the use of Edge computing/TinyML platforms enable ML in the greenhouse, avoiding cloud-based services and without overly penalising the

overall power consumption of the greenhouse. The main contribution of the paper are the following:

1. A comprehensive analysis is provided to assess the limitations of different TinyML/Edge computing platforms to serve a real operating environment such as a greenhouse. Several datasets introducing climate variability typical of semi-arid climates are defined to train and validate the predictive models evaluated.
2. Two different ANNs, namely the multilayer perceptron (MLP) and the convolutional neural network (CNN), are used, evaluated and parameterised to predict the internal temperature of an operating greenhouse. These ANNs are adapted to reduce their memory footprint by testing different neural network architectures, adjusting their hyperparameters, reducing the accuracy of the variables involved and migrating each model to the target low-power devices.
3. The process of migrating these workloads to the targeted edge computing platforms, including an Arduino family microcontroller, is shown.
4. A performance and energy consumption tradeoff of these platforms is under study, showing clear advantages to microcontrollers for these lightweight workloads.

The paper is organised as follows. Section 2 describes a set of related works on prediction of climate variables with different machine learning techniques executed in edge and tiny environments. Section 3 describes the datasets used, as well as the models and methodology followed in the development of the work. Section 4 shows the results and the analysis and discussion of the results. Finally, the conclusions and future work are shown in Sect. 5.

2 Related works

Weather forecasting has emerged as an important topic of research in the last decades. Particularly the prediction of climatic variables is a topic of special relevance in agriculture. In outdoor agriculture and greenhouses, knowing the parameters of certain climatic variables helps farmers make decisions that optimise their resources. Since time is nonlinear and dynamic, the techniques used must also be nonlinear [12]. In the case of this work, we have focused on neural networks to predict temperature. Moreover, given that the agricultural world, especially small companies, cannot make large investments in technology, we propose the study to evaluate the performance and consumption of low-cost devices. In this section, we briefly study other works that have made predictions of climatic variables, considering neural networks and their adaptation to run in edge environments.

In [13], a technique is presented for predicting daily minimum, average and maximum temperature using three types of artificial neural networks, namely multilayer perceptron, recurrent neural network, and convolutional neural network. The best temperature prediction result was obtained using a convolutional neural network. Another work that uses convolutional neural networks is presented in [14]. The authors present a hybrid model of convolutional neural networks and recurrent

networks to predict temperatures. The model uses daily data from mainland China and obtains an error of less than 1 degree Celsius. In [15] Jung et al. propose a comparison of various neural network techniques, with different hourly granularities to predict climate variables, such as humidity and temperature, inside the greenhouse. The best results were obtained with a combination of Recurrent neural networks and Long short-term memory. These works, although related to temperature prediction using neuronal networks, do not take into account the adaptability of these models to be executed at the edge of network.

In [16] the authors present an approach to improve the management of greenhouses and predict their internal variables using recurrent neural networks executed and adapted to the edge computing environment. Among the variables they predict is the air temperature. The proposed model allows them to anticipate the reading of the greenhouse sensors locally. Another work where they also predict air temperature using neural networks and edge computing is presented in [17]. The authors forecast the anticipated temperature to determine when a frost will occur. In the study, the authors make a comparison between running predictions in the cloud and in edge environments. The conclusions indicate that the results in edge environments are satisfactory and that for agricultural environments where connectivity cannot always be ensured, edge environments are efficient. In [18], authors proposed the evaluation of three different types of neural network architecture, using different values of the sliding window associated with the input data run at the edge to predict the indoor temperature of a greenhouse. In [19], authors presented a study analysing advances in work using edge computing, demonstrating its effectiveness in facilitating data analysis, future prediction and decision making at the edge by avoiding the transfer of large volumes of data in classical systems. Also in [20], a study was made for time series data forecasting, where the different factors that influence the energy consumption of smart metres are studied, analysing several issues, including temperature forecasting. This study demonstrates the superiority of LSTMs over other models. From the point of view of efficiency and speed, in [21], the authors used TPU accelerators at the edge, analysing three different types of TPUs. These models allowed for faster and less time-consuming evaluations. In [22], they proposed an adaptive CNN for low power consumption edge computing devices. They proposed a novel adaptive architecture that used an output block predictor to choose the base architecture for inference, providing similar or superior performance to the classical architecture. Another work on Edge computing is presented in [23], where the authors connected and managed IoT devices to analyse information from strawberry crops to detect diseases in these crops. In [24], authors used a model for workload forecasting with adaptive sliding window and use temporal correlation by means of an adaptive sliding window algorithm in order to achieve higher accuracy with lower overhead. In [25], taking into account that the use of cloud applications is becoming more and more popular and specifically through containers, they presented a model for the calculation of container similarity for scenarios presented in the cloud, providing a new solution in workload prediction models for this type of containers.

Regarding TinyML, there are not many studies available about its use to predict climate variables, yet. In [26], authors presented the design of a tiny deep neural network to predict atmospheric pressure, embedded in a microcontroller. The tiny

Fig. 1 Targeted operational greenhouse located at Murcia (Spain)

neural network approach presented was compared with the results of a non-tiny neural network, obtaining similar results. Furthermore, experiments demonstrated the performance of the approach in a real environment.

Analysing the related works, the reader can figure out that there is no assessment, analysis and comparison of the performance of climatic variables in both edge and TinyML platforms. This is one of the main novelties presented in this work, together with the discussion and measurement of the different energy consumption.

3 Materials and methods

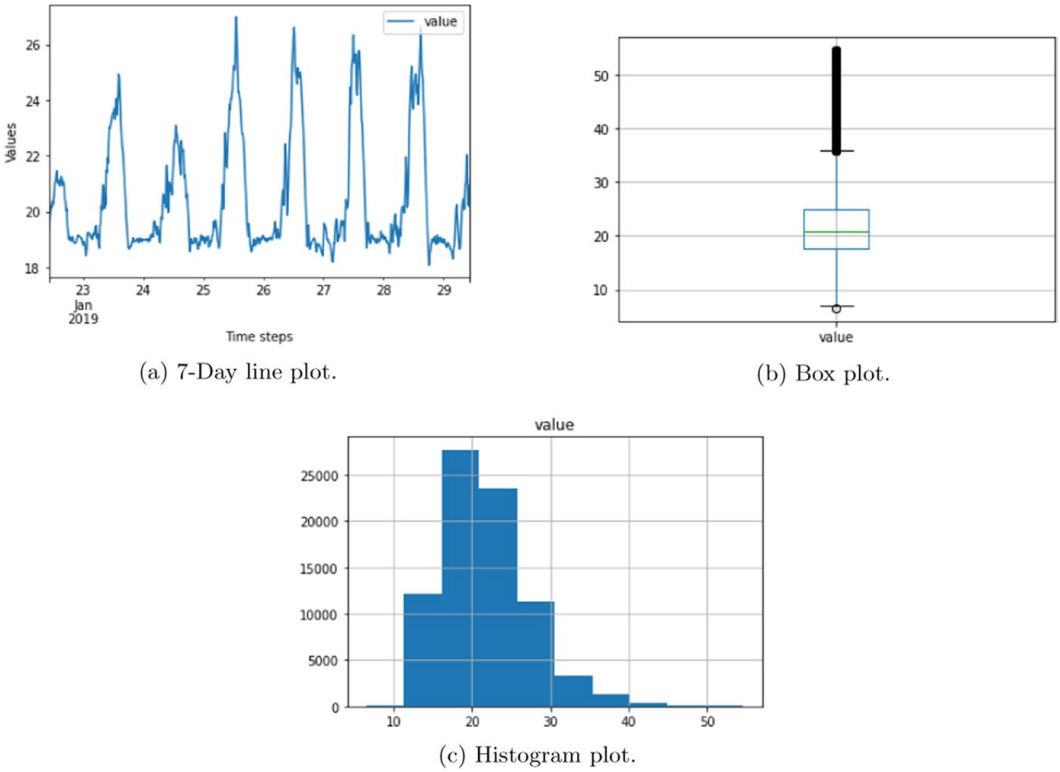
3.1 Operational greenhouse

Figure 1 shows the ETIFA's operational smart greenhouse targeted for this study. ETIFA[1] is part of a group of companies with more than 40 years of experience providing agricultural services and technology, manufacturing and innovating to adapt to the continuous demands of the agricultural market. ETIFA's greenhouse is placed in Murcia (South-eastern Spain); a semi-arid region where water is very scarce resource (e.g. the average annual rainfall in the last years is 132 mm) and average annual temperature is around 23° C. This greenhouse has a surface area of 50 m^2 and operates with the NUTRICONTROL OPTIMUM® system for climate control and fertigation. NUTRICONTROL is a R+D company that offers solutions for the analysis, design, manufacture and marketing of control equipment for irrigation and climate automation in greenhouses and outdoor irrigation installations. The OPTIMUM® system consists of a data logger composed of a server system and several input/output connections to plug-in several sensors and actuators, including

[1] https://www.etifa.com/.

Table 1 Dataset description

Datasets	Start date	End date	# Instances
DS-15	11-12-2018	23-03-2021	80018
DS-60	11-12-2018	23-03-2021	20005



(a) 7-Day line plot. (b) Box plot.

(c) Histogram plot.

Fig. 2 Exploratory data analysis (EDA) for the 15-min time series

temperature, humidity, radiation, wind speed, just to name a few. Of particular interest to us is the temperature of the air inside the greenhouse as this is the most important variable for climate control. This variable is measured every 5 min in the ETIFA greenhouse, providing near-real time (NRT) continuous measurements to take actions that can increase/decrease the greenhouse temperature to reach the ideal temperature of the crop being grown.

3.2 Dataset

Table 1 summarises the description of the datasets that have been used to carry out the temperature prediction. It shows the start date of the data, the end date and the number of instances contained in each dataset. Each dataset contains the temperature values each 15 or 60 min of a greenhouse between the indicated dates.

As the sensors return data approximately every 2–5 min, to obtain the 15-min and the 60-min datasets, an aggregation of the data has been made, in which each value corresponds to the average of the values belonging to that time interval.

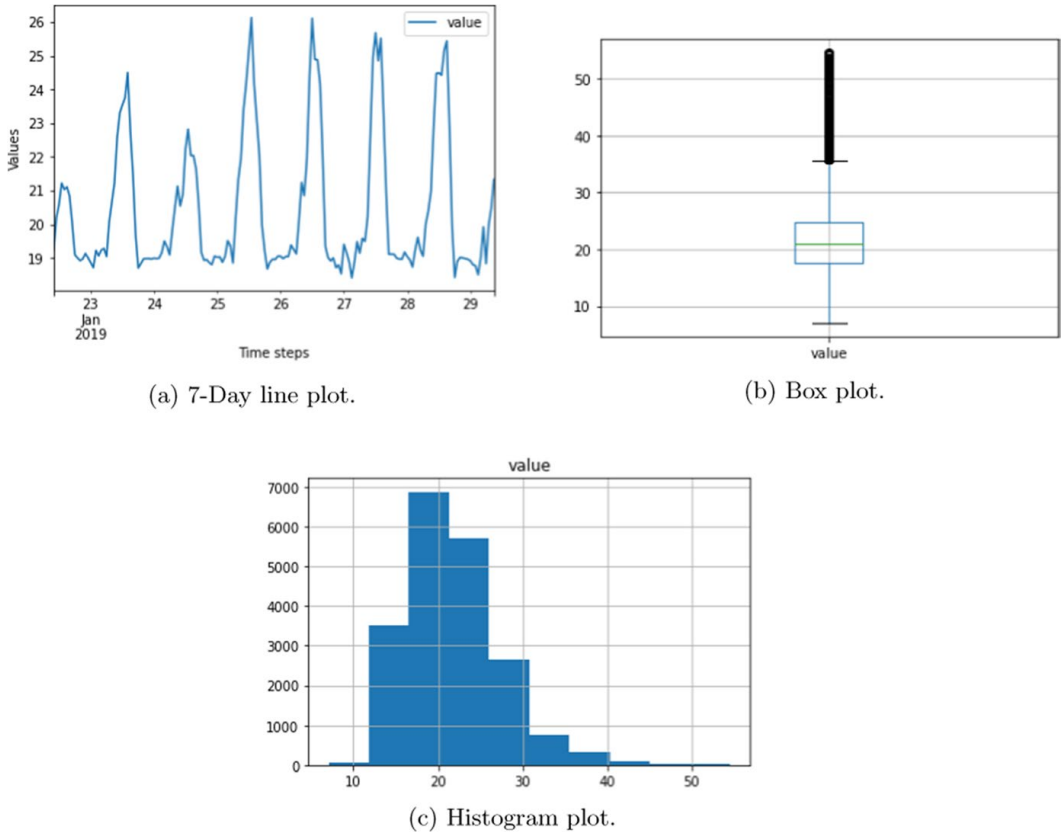


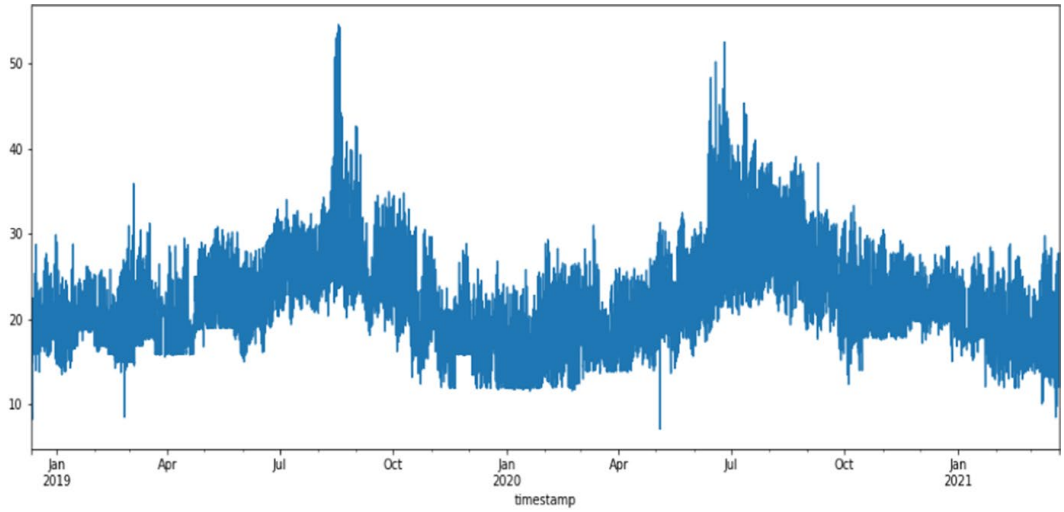
Fig. 3 Exploratory data analysis (EDA) for the 60-min time series

Figure 2 shows an exploratory data analysis of 15-min dataset. This dataset includes 80,0018 different values with an average mean of 21.689° C and standard deviation of 5.726° C. The maximum average value reaches up to 54.620° C. The time series of the internal greenhouse temperature is shown in Fig. 2a which illustrates an example of the 7-day time series. Moreover, the box-and-whisker plot (see Fig. 2b), and the distribution of its values (see Fig. 2c) are shown.

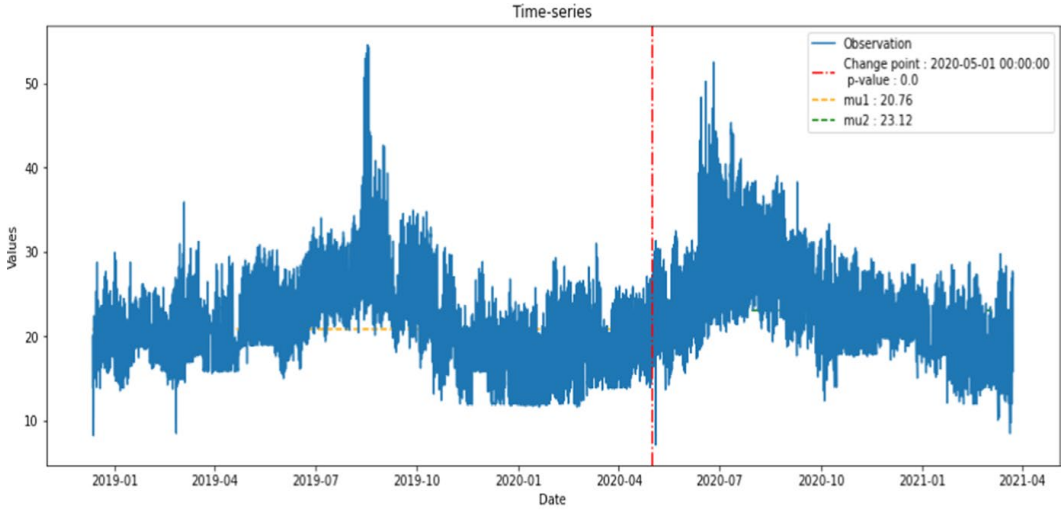
Similarly, Fig. 3 shows the description of the 60-min dataset. It includes more than 20,000 records with a mean of 21.689° C and a standard deviation of 5.705° C. An example of the 7-day time series is also shown (see Fig. 3a) where a smoother data was appreciated. Finally, the box and whisker plot (see Fig. 3b), and the distribution of its values (see Fig. 3c) are shown.

Finally, Fig. 4 shows the complete time series (see Fig. 4a) as well as the result of applying Pettit's homogeneity test [27] (see Fig. 4b), in which it can be seen that the time series is not homogeneous, with a possible cut-off point on 01-05-2020, dividing the time series in two: the first part whose mean is 20.76° C and the second part whose mean is 23.12° C.

Once the data has been collected as described above, it has to be prepared for training and inference of ANN models. Specifically, we focus on the prediction of a time series, in this case, the internal greenhouse temperature. Equation 1 shows the input and out layout of our dataset to cast and adapt the data and to be trained by ANNs models.



(a) Full time serie.



(b) Results of Pettit's homogeneity test.

Fig. 4 Homogeneity of the time series

$$\begin{aligned} &input \rightarrow output \\ &[t_1, t_2, t_3, ..., t_n] \rightarrow [t_{n+1}, t_{n+2}, t_{n+3}, ..., t_{n+m}] \end{aligned} \tag{1}$$

where n is the number of past elements that the network will have to infer the next value. This parameter is often referred to as look-back. In addition, m defines the number of instances to be predicted by the model. It should be noted that depending on the input dataset (DS-15 or DS-60) each value to be predicted means a different time granularity. For example, if the model predicts the next 3 h of greenhouse interior temperature, with DS-15 it means generating 12 values (i.e. $m = 12$). However, if the model has been trained with DS-60, the 3 h forecast means generating 3 values (i.e. $m = 3$). Figure 5 shows how the inputs are selected for each of the iterations in the DS-60 case, where each of the three inputs corresponds to one hour. The output

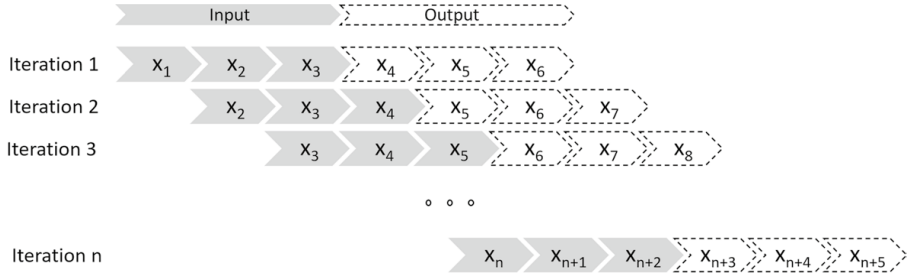


Fig. 5 Inputs and outputs diagram to train ANN models

obtained is the following three hours. For the DS-15 case, the scheme is the same but instead of having three inputs and three outputs, there are twelve.

Finally, the last data transformation performed on the dataset to adapt it to ANNs is the separation into training data and test data, i.e. data that will be used to make the model learn (training) and data that will be used to validate the accuracy of the model (testing). The percentages associated with each subset of data are 70% for the training data set and 30% for the testing data set.

3.3 The ANN models

This section describes the evaluated ML models. They are ANNs-based models relatively simple as they have to fit in very reduce memory space. The models chosen were the multi-layer perceptron (MLP) and the convolutional neural network (CNN), both very simple models that are supported by the *tflite* conversion that allow the execution on the microcontroller. A neural network is an ML model that mimics the way a set of biological neurons works. Although its use in classification is more widespread, it can also be used in regression models. A perceptron simulates an artificial neuron, so that it represents a simulation of a logistic regression. When a group of perceptrons per layer (MLP) is joined together, it is known as an artificial neural network (ANN). The MLP used in this work is composed of three layers: input (receives the input features), hidden (processes the inputs) and output (produces the output). In the learning process, each layer adjusts its weights to relate inputs to outputs, because they use an activation function that allows them to learn nonlinear properties in the network [28, 29]. The second model under study is a convolutional neural network (CNN). The CNN models are used in different applications and domains, where they are most frequently used in imaging for classification. However, they are also used in regression, where they can be used using time series by transforming the data to adapt them to the inputs of the convolutional network. A CNN is made up of blocks of filters, which, through convolution operations, allow the relevant features to be extracted from the input. One of the advantages of CNNs over conventional neural networks (ANNs) is the automatic learning of the filters, so that the necessary and most relevant features are obtained from the input data [30, 31].

ANN models are supervised methods that consist of two main steps. The first is training where a known data set, i.e. where the input and output are known, is

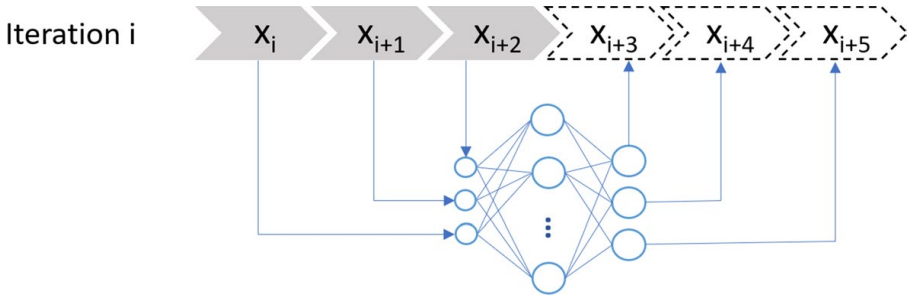


Fig. 6 Model used for the MLP scheme for the case of 3 h inputs The MLP has three layers (input, hidden and output), where the hidden layer learns the model that allows the output to be predicted from the input

used to fit the model with a custom parameters. In our case, we introduce the different temperature tuples prepared to serve as input (i.e. t_1, ..., t_n) to the different networks. The ANNs generate an output which is compared with the previously known output (i.e. t_{n+1},..., t_{n+m}).

The ANNs compare the real output and the forecasted output, re-evaluate the error value and update the weights of each neuron in the neural network layers depending on how correct or incorrect the forecast is, so that the neural network is adjusted in the training process, improving the performance of the task it is learning.

The inference process is different from the training process. In inference, the layers of the neural network are not re-evaluated and adjusted. Simply, knowledge from a neural network model that has already been trained is applied and a result is inferred without readjusting the weights of the neural network.

In our case, as said before, we use two different models, MLP and CNN. This models have a custom configuration (a.k.a. hyperparameters) in order to improve the performance of it forecasting.

The most important hyperparameters for a MLP in time series regression problem are:

- Units: Number of neurons used in one layer of the model.
- Optimizer: It is a function that optimises the learning of an ANN, updating its neurons weights depending on the evaluation error
- Learning rate: It allows the speed of adaptation of the model to the problem to be established.
- Loss function: Function used for evaluate the error of the model in each epoch.

Regarding the MLP, we have used a hidden layer with 100 units, an Adam optimizer, a learning rate equal to 0.000001, and MSE (Mean Squared Error) as a loss function. Also, for training, the model is using a total of 100 epoch for 15-min dataset and 200 epochs for 60-min dataset. Figure 6 shows the scheme used for the MLP network for the 3 h case using 60-min dataset, where the input is passed directly to the MLP network and the prediction of the next three hours is obtained. For the 15-min dataset case, the scheme would represent twelve inputs and twelve outputs.

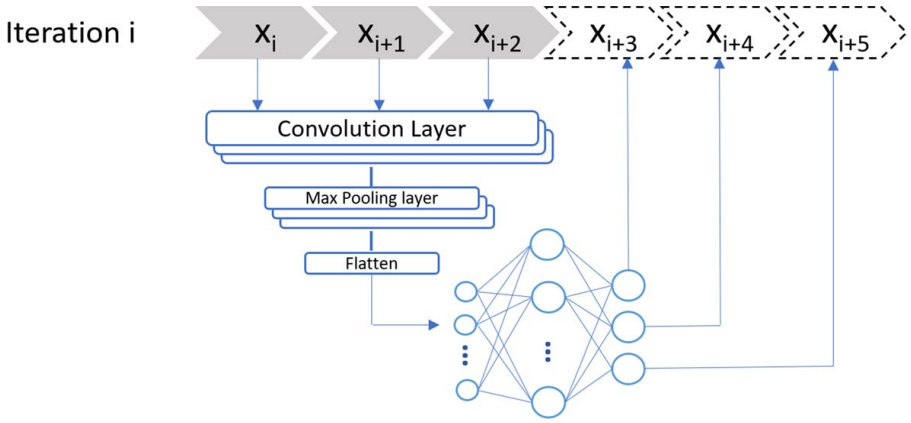


Fig. 7 Model used for the CNN scheme for the 3 h input case, where a new feature vector is obtained due to the convolutional layer. In this way, the hidden layer of the network learns to generate the output from these new features provided by the convolutional layers, unlike the previous case where only an MLP is used with the direct input

The most important hyperparameters for a CNN in time series regression problem are:

- Filters: Sets the dimension of the output space.
- Kernel size: Sets the width and height of the 2D convolution window.
- Padding: Add irrelevant pixels when a window is smaller than standard window.
- Activation: Function to define how to transform the weighted sum of the input into an output for a node or nodes in a network layer.
- Pool size: Window size over which to take the maximum.
- Optimizer: It is a function that optimises the learning of an ANN, updating its neurons weights depending on the evaluation error.
- Learning rate: It allows to control how fast a model adapts to the problem.
- Loss function: Function used for evaluate the error of the model in each epoch.

Regarding the CNN, we have used a convolutional layer with 16 filters, [2, 1] of kernel size, "same" padding, "ReLU" as activation function, [2, 1] of pool size, an Adam optimizer, a learning rate equal to 0.00001, and MSE (mean squared error) as the loss function. Also, for training, the model is using a total of 50 epochs for 15-min dataset and 20 epochs for 60-min dataset. Figure 7 shows the scheme used for the CNN network, where the input is passed through a convolutional layer to obtain a feature vector that is used as input to the fully connected neural network. This figure represents the example of 3 h and prediction of the next three hours with 60-min dataset. For the 15-min dataset case, the scheme would represent twelve inputs and twelve outputs.

As a framework we use TensorFlow (developed by Google), which allows us to build machine learning models.

3.4 TensorFlow Platform used to build the ML models

TensorFlow is a powerful open-source deep learning framework that can run machine learning models on various devices, including Raspberry Pi and Nvidia Jetson. Both Raspberry Pi and Nvidia Jetson are popular choices for building intelligent embedded systems and IoT devices due to their low cost, small form factor, and powerful processing capabilities.

When using TensorFlow on a Raspberry Pi, developers can take advantage of the framework's ability to perform complex machine learning operations, such as image classification and object detection, on the device itself. This allows for real-time processing of data, making it possible to build intelligent applications that can respond to their environment in real-time. Additionally, TensorFlow supports a wide range of hardware and software platforms, including Linux and Android, which makes it easy to use on the Raspberry Pi. This can be especially useful for projects that require low-level control of the hardware, such as robotics or home automation systems.

Similarly, Nvidia Jetson boards are also powerful devices that can run TensorFlow models with high performance. Jetson boards are based on the NVIDIA CUDA architecture, which is specifically designed for running deep learning workloads. This makes them ideal for applications such as computer vision, object detection, and image recognition. The Jetson boards also have a powerful GPU and a large amount of memory, which allows them to handle large and complex models.

Using TensorFlow on both Raspberry Pi and Nvidia Jetson can be a great way to take advantage of the framework's powerful capabilities while still keeping the costs and power consumption low. Additionally, TensorFlow allows developers to use pre-trained models and a library of powerful algorithms to train their own models, making the development process faster and more efficient. It also allows developers to deploy the models on the device, which can make the application more efficient, reliable and secure.

TensorFlow Lite is a lightweight version of TensorFlow. It is designed to help developers deploy machine learning models on mobile and IoT devices with limited computational resources, such as Arduino Nano devices. It has been optimised for these types of devices, making it possible to run models on devices with limited memory and processing power.

One of the key features of TensorFlow Lite is its ability to convert pre-trained TensorFlow models into a format that can be run on mobile and IoT devices. This conversion process, known as "model quantization", reduces the size of the model and enables it to run faster on these devices. TensorFlow Lite also includes a number of other performance optimizations, such as support for hardware acceleration, to further improve the performance of models on these devices. Additionally, TensorFlow Lite provides a user-friendly API that makes it easy for developers to integrate machine learning into their mobile and IoT applications.

3.5 Training and deployment of ML models in edge and Tiny ML platforms

As mentioned above, ANN models are supervised models, so they are run in two different steps: training and inference. Training is the most time-consuming step and is a process that, to date, cannot be carried out on microcontrollers. Therefore, the training process for the models targeted in this paper is carried out on an high-performance computing (HPC) platform and it is exactly the same procedure applied to every ML problem, i.e. pre-processing and transformation of the data, development of the ML model and then training with the dataset. Once trained, the model is tested with the test set and if necessary, the model is re-trained with different parameters.

Once the ANN models have been trained, they need to be transferred to the platform where the inference is performed. This paper targets four different computing platforms (see Fig. 8). Three of them can be classified as edge/fog computing platforms, i.e. Raspberry PI Model B and two platforms from Nvidia Jetson family. The Raspberry PI 4 Model B is a single-board computer (SBC) that is extensively used in IoT infrastructure. It has a Quad core Cortex-A72 (ARM v8) 64-bit SoC, running at 1.5GHz and endowed with 8GB LPDDR4-3200 SDRAM. Regarding the Nvidia Jetson edge computing devices, we focus on the Nvidia AGX Jetson Xavier with 8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3, 512-core Volta GPU with Tensor Cores and 16GB 256-Bit LPDDR4x running at 137GB/sec, and Nvidia Jetson Nano that has 5-core ARM Cortex-A57 MPCore CPU, 2MB L2, 128-core Maxwell GPU and 4GB 64-Bit LPDDR4 running at 25.6 GB/s.

The last device under study is the Arduino Nano 33 BLE Sense that can be categorised as TinyML device since combines small factor, environment sensing and allows ML models to be run using TinyML and TensorFlow Lite. It is build upon the nRF52840 64MHz microcontroller, the memory is 256 KB SRAM, 1MB flash, and runs on ARM Mbd OS. It is important to note that the main way to connect to this Arduino device is via Bluetooth Low Energy although it is also equipped with some sensors to detect audio, colour, humidity, temperature, motion, proximity and more.

As can be seen, the platforms have very different characteristics, with large differences in computational capabilities. Therefore, it is necessary to make a series of modifications to the artificial intelligence models so that they can be adapted to the characteristics of all platforms. In particular, to achieve this objective: (1) different configurations of hidden layers of the models have been studied until we found a configuration that was compatible with all deployment platforms, (2) an optimal hyperparameter configuration has been searched to maximise the quality of the results obtained, minimizing the number of trainable parameters of the model and (3) the model has been translated into different programming languages depending on the platform on which it was going to be executed, for example, for Arduino and Raspberry it has been necessary to build the model in C while for the Nvidia Jetsons Python has been used as programming language. In this way, models are obtained with a reduced number of trainable parameters, with a reduced memory space and with low computational requirements that allow them to be executed in each and every one of the platforms that are the object of this study.

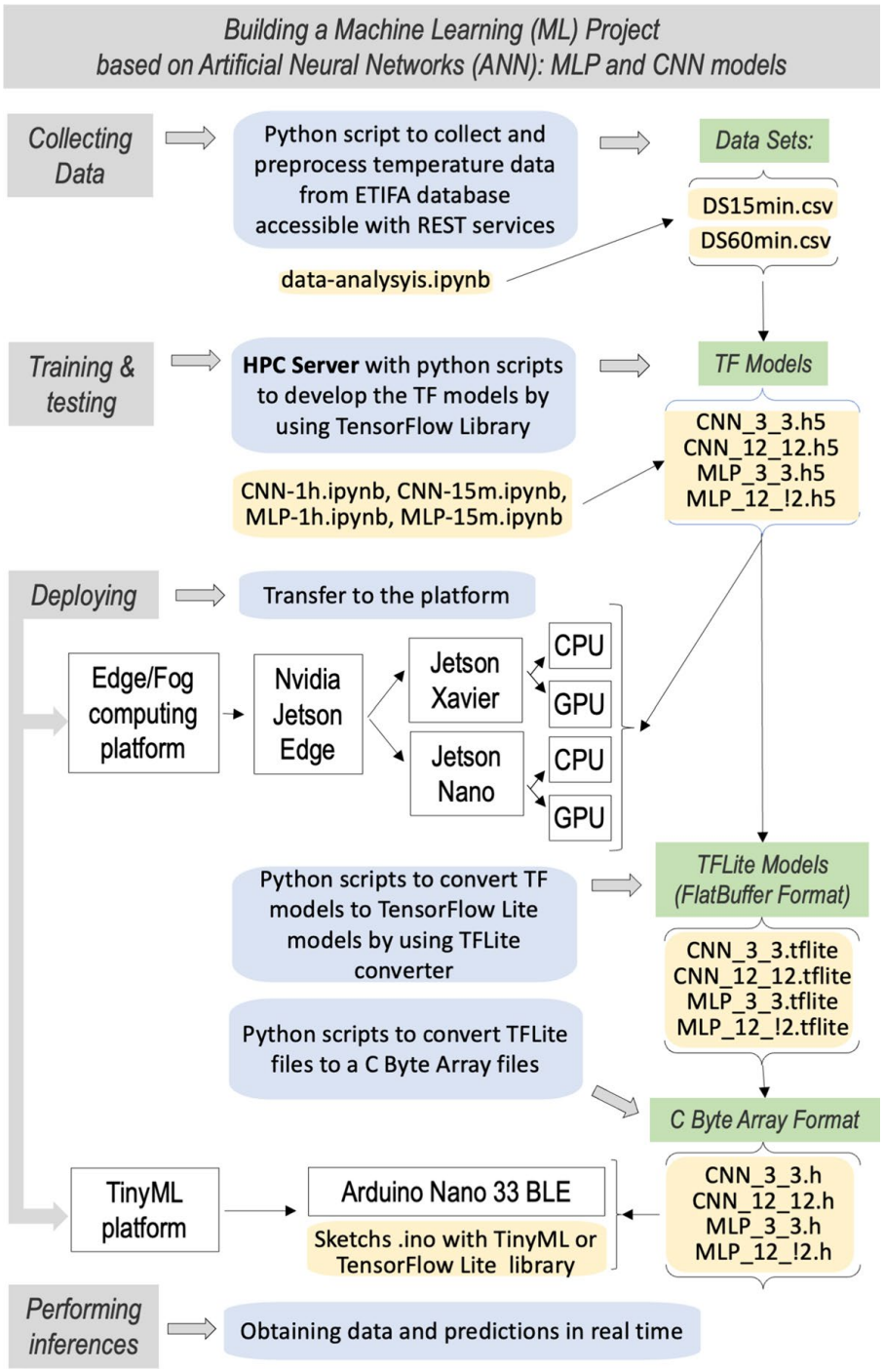


Fig. 8 Building a Tiny ML project

To deploy the models on edge computing platforms, it is necessary to first train the models on HPC computers and, using the library's predefined functions, export them to a *.h*5 file. Once the AI model has been exported to a file, it is necessary to move the model file to the edge computing platform on which it will run, load it (also using the library's predefined functions) and finally perform the inference. For the GPU versions of Nvidia's Jetson family, the procedure is similar with the

difference of the value of an environment variable, `CUDA_VISIBLE_DEVICES`, whose value was "" to force execution on CPU or "0" to force execution on the GPU available on the system.

Deployment on the TinyML platform is not so straightforward. The trained model must be converted to TensorFlow Lite [32]. TFLite is the framework for deploying ML models on mobile, microcontrollers and other edge devices that has a reduced memory space. FlatBuffers is how a TFLite model is represented, where its extension is.tflite. It allows for a special, portable and efficient format. TFLite has advantages over the buffer model format of the TensorFlow protocol, featuring faster inference because it accesses the data directly without an additional parsing step, as well as a reduced memory size (small code). These features allow TFLite to run efficiently on devices with limited processing and memory resources. A TFLite model can be generated in several ways. First, using an existing TFLite model available in the TFlite SDK.[2] Second, creating a TFLite model, using TFLite Model Maker to create a model with your own custom dataset. It is noteworthy to highlight all models already contain metadata. Third, converting a TF model to a TFLite model by using the TFLite converter. During conversion, several optimisations such as quantization to reduce model size and latency with little or no loss of accuracy can be applied. Models developed buy this last option no include metadata. In our case, we chose the third option to get the TFlite model to load on the Arduino. It is important to note that not all TF models can be converted to TFLite models that is one of the reason why this work focuses on MLP and CNN models and not in other models such as LSTM. Finally, it is necessary to convert the.tflite file to a byte array in C. in order to run the model on an Arduino.

The last part is represented by the inference process in the device to be used. The edge computing devices receive the data through REST API server, and once they have received the input data, they generate the forecast values. On the other hand, the Arduino receives the information via bluetooth.

4 Evaluation and discussion

This section shows the performance, energy and quality results obtained on all targeted devices; i.e. Arduino Nano 33 BLE Sense, Raspberry Pi 4, Nvidia Jetson Nano and Nvidia AGX Xavier by executing MLP and CNN models previously presented in Sect. 3.3. The models developed are named with the following abbreviations: *MLP15m* and *MLP60m* for the MLP model using 15 and 60 min data aggregations, and *CNN15m* and *CNN60m* for CNN models using the same data aggregations. We refer the reader to Sect. 3.2 for insights on the dataset construction.

Figure 9 shows the execution time for inference of MLP and CNN models on all targeted platforms. The Arduino Nano 33 BLE Sense is the worst performance platform as expected. The computational differences of the Arduino against the rest of the platforms are clearly lower in the MLP inference, where a maximum difference

[2] https://www.tensorflow.org/lite/examples.

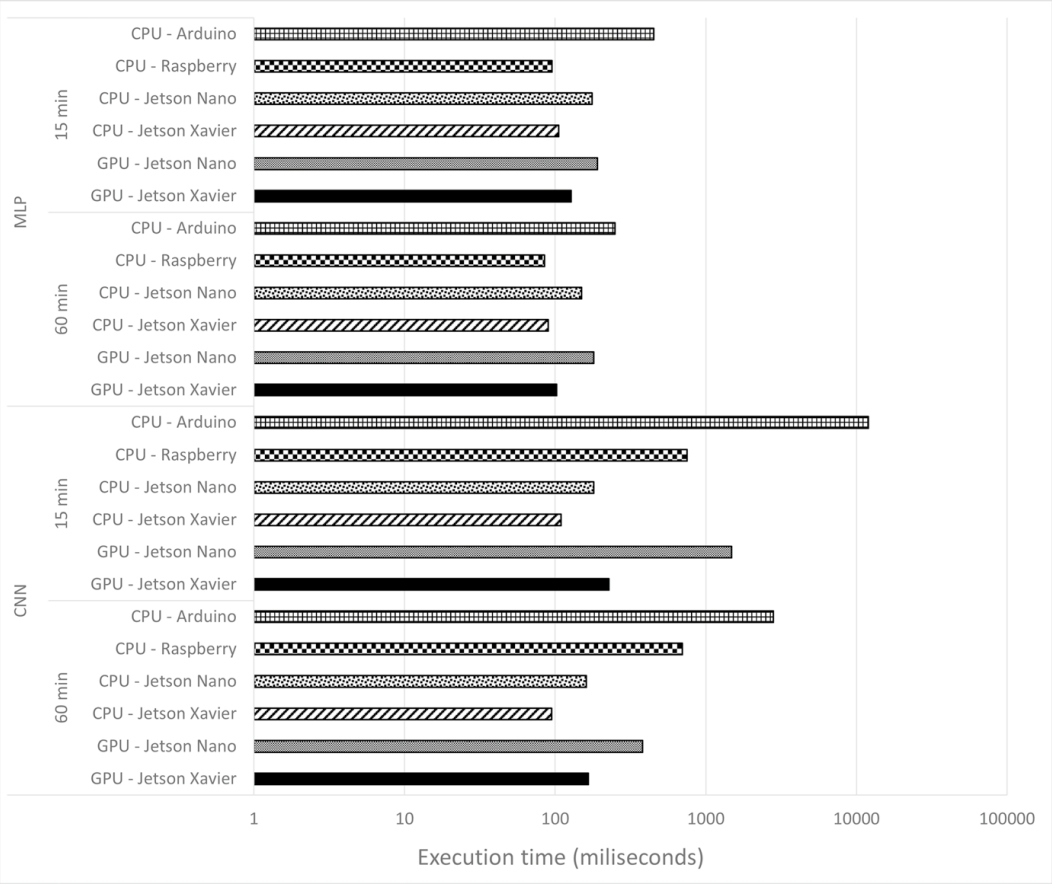


Fig. 9 Execution time (milliseconds) in logarithmic scale of ML models on different devices

of 4.7X speed-up factor is reached, than in the CNN inference, where a speed-up factor of more than two orders of magnitude is reached (110X speed-up factor between Jetson AGX Xavier and Arduino). MLP model inference is computationally lighter than CNN model inference because it has fewer layers and the operations carried out are much simpler than MLP. It is important to note that Fig. 9 is shown in logarithmic scale due to the large difference in performance of the platforms under study. Indeed, it is possible to clearly appreciate, for example, the great difference in time between running the ML workloads on Jetson Xavier CPU or the Arduino one.

Computational differences between Arduino and other platforms are also reduced by the number of elements to be predicted. In the 60-min dataset, the computational difference is reduced 4 times in terms of speed-up factor for *CNN60m* compared to *CNN15m* and 1.5 times for *MLP*60*m* compared to *MLP*15*m*. Overall, the number of elements to be generated in the 60-min dataset prediction is 4 times smaller than in the 15-min dataset prediction.

It is also important to note the negative impact of the use of GPUs in this context, reported by the Jetson family. ML models under study are extremely lightweight to be executed on TinyML platforms such as Arduino ones. Therefore, the computational needs of these workloads are not large enough to fill all GPU resources, penalising the execution time of the inference. This penalty is also highlighted in the inference of the CNN model that is clearly affected by memory latency, as CNN

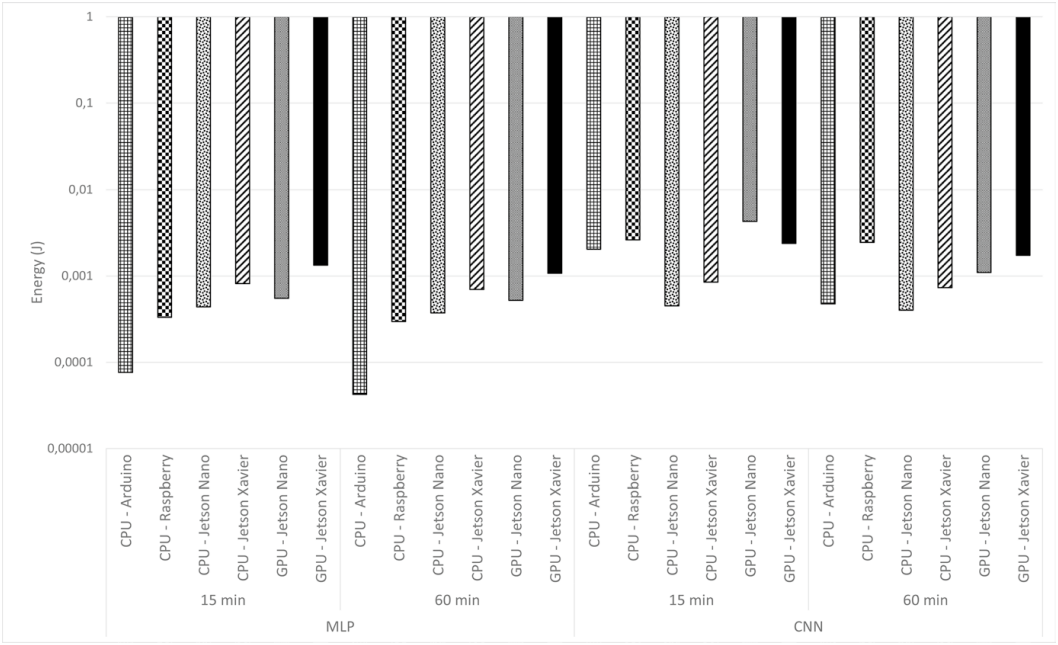


Fig. 10 Energy consumption (Joules) in logarithmic scale of ML models on different devices. Note that below 1, the larger the bar the lower the energy consumption value

has a higher number of memory access. Actually, this memory pressure is also shown by determining the best performing platform, which for MLP inference is the Raspeberry 4 with little difference (5–10%) to the Jetson AGX Xavier which has better technical specifications. However, CNN model inference performs better in the Jetson AGX Xavier by a wide margin since it benefit from the higher memory bandwidth.

Figure 10 shows the energy consumption in Joules of the different platforms analysed by running the ML model inference. Again, it is important to note that the Y-axis is on a logarithmic scale and therefore below axis 1, the larger the bar, the lower the value of energy consumed. To obtain these figures, the Microchip's PAC1934 power metre is used that provides instantaneous power consumption, energy accumulation, etc., by plugging into USB type-C bus used for power supply [33]. The power consumption is tracked when ML models are running on the targeted devices and thus making predictions. Moreover, Fig. 10 shows the maximum values reached during the execution of each ML model. Figure 10 is also shown in logarithmic scale given the wide range of values, as was the case in Fig. 9

Regarding the power consumption (Watts) figures, differences are quite large; the Arduino Nano consumes only 0.17 W, defeating Raspberry PI by a wide margin as it consumes 3.5 W. Regarding Nvidia Jetson platforms, there are also some important differences. The idle power consumption of Jetson Nano is in the range of 1.3 W and 1.6 W. The power consumption of CPU-based MLP and CNN models is in range of 2.4 W and 2.6 W. Whenever the GPU is switched on, the power consumption of the Nvidia Jetson Nano increases by a factor of 13.79%, reaching up to 2.9−3.1 W. The power consumption of the Jetson AGX Xavier with GPU enabled reaches up to 10.15 W while with only executing the CPU version consumes 7.77 W. With this

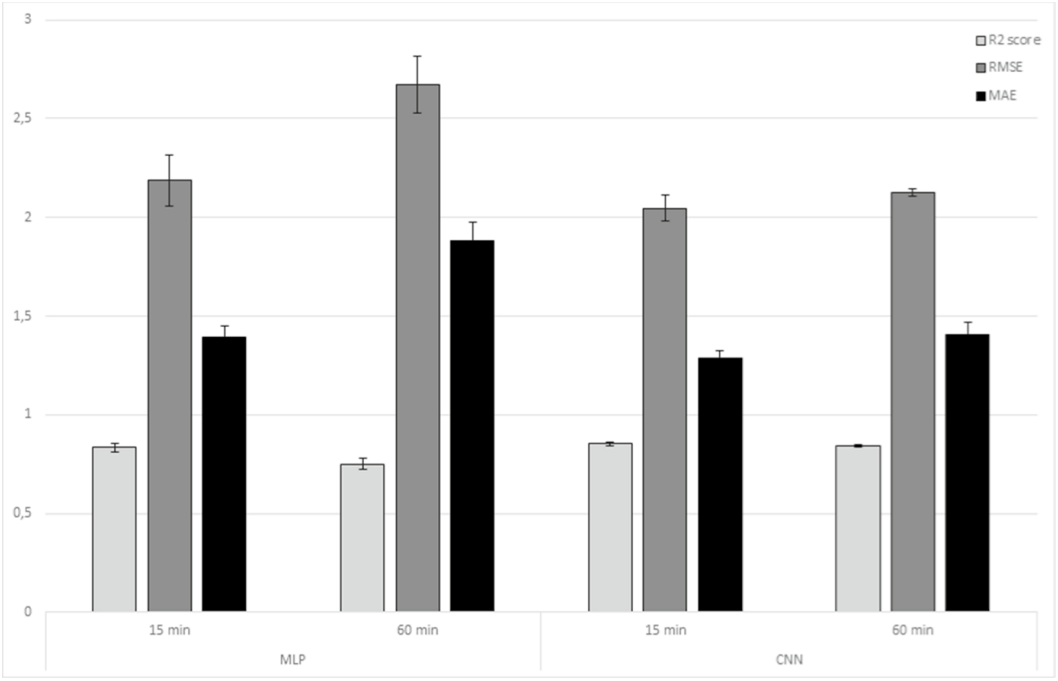


Fig. 11 Metrics (R^2 score, RMSE and MAE) of ML models on different devices. The bars show the mean of the metric in question for the different platforms while the vertical line at the top of the bars shows the standard deviation

in mind, the Arduino is by far the most energy efficient platform (about 1 order of magnitude), followed by the Raspberry PI, the Jetson in CPU mode and finally, the least energy efficient platform for these workloads is the Jetson family, using the GPU. As discussed above, the small difference in performance between the platforms is not enough to hide the energy consumption of the platforms. The Arduino would have a power consumption of 0.17 W. Once the instantaneous consumption of the platforms was calculated, the energy was calculated in a straightforward way, multiplying the execution time in seconds, and thus being able to obtain a metric that shows the best model-device combination in the trade-off between execution time and energy consumption. Actually, according to the energy figures shown in Fig. 10, the most energy efficient model-device combination is the *MLP*60*m* and the *CNN*60*m*, running on the Arduino microcontroller. The Arduino microcontroller consumes less power than the Raspberry although it takes longer to perform the predictions. This result is exactly what was expected due to the hardware difference between the two devices and shows perfectly why TinyML, and more generally bringing ML to microcontrollers, is so important and convenient.

Finally, Fig. 11 shows the accuracy of the models by forecasting the internal temperature of the greenhouse for the next 3 h. The metrics used in this evaluation are as follows. The root mean squared error (RMSE) is the square root of the average of squared errors. The mean absolute error (MAE) is the average absolute difference between the observed and the predicted value. These two metrics are measured in Celsius degrees as they would be errors in the temperature prediction. The lower the error, the better the result of the technique obtained.

Finally, the coefficient of determination R^2 is used to determine the goodness of the model. This value is estimated at 0 and 1, with values close to 1 being a better fit and values closer to 0 a worse fit. Results in Fig. 11 have been calculated for both the MLP and CNN techniques, dividing into 15-min and 60-min datasets, and averaged them for each of the devices used. In addition, the standard deviation obtained for each metric is shown at the top of the graph.

Analysing the results, the best model on the RMSE value is *CNN*15*m* as expected. The CNN model is more complex ANN than the straightforward MLP and can theoretically better capture the behaviour of the time series. Moreover, the dataset with 15-min measurements has a larger size, so the models have more data to improve the training. In general, RMSE results show that the mean deviation from the forecast values is between 2° C and 3° C. MAE results, however, show that the mean difference between the forecast and the observed temperature is between 1 ° C and 2 ° C. Since the RMSE-MAE difference is not large enough, it means that large errors are unlikely to have occurred. As for the R^2 results, *CNN*15*m* and *CNN*60*m* reached a value of almost 0.85, which means that the predictions of these models are quite accurate, while *MLP*15*m* is 0.8 and *MLP*60*m* has the lowest score of 0.7. The difference in terms of evaluation metrics between the two models is obvious but it does not necessarily mean that MLPs are not a valid choice because, as stated before, they are lighter and their execution is faster. Of the three metrics shown in Fig. 11 and with respect to quality results, *CNN*15*m* shows the best performance, followed by *CNN*60*m*, *MLP*15*m* and *MLP*60*m*.

It is important to highlight the great difference between the 15-min and 60-min datasets. As we have already mentioned, the CNN technique obtains the best result compared to MLP, the models obtained for the two datasets have a similar behaviour. Better results are always obtained with the 15-min dataset than with the 60-min dataset. The reader may think that the models are over-fitted or underfitted. But the reality is that the 15-min dataset aggregates a lower granularity, so the deviation of the 15-min temperature is smaller than it does for 60-min dataset. This leads to a lower prediction error in 15 min than in 60 min and this also leads to a better system tuning. The *MLP15m*, *MLP60m* models are run with the same parameters optimised to the average case. The same is valid for the models *CNN15m*, *CNN60m*. Thus, we can conclude that when predicting temperature it will always be better to have lower granularity, not only for the improvement of the results, but from an agricultural point of view, to be able to act and control the temperature in short periods of time.

With all of the above in mind and taking into account the performance, energy and quality figures, we are tempted to highlight as the best hardware-software combination the *CNN*60*m* model running on the Arduino microcontroller. The *CNN*60*m* model offers similar quality numbers to the *CNN*15*m* model but the energy used when running that model on the microcontroller is lower. However, there are other factors to consider and these depend on the particular application case where the infrastructure is deployed. In an application such as smart greenhouse management, the energy consumed is probably an important aspect

while the accuracy of the model is not so-critical as long as the error is under certain threshold. MLP models executed on the Arduino microcontroller are a better choice as it offers almost an order of magnitude lower energy consumption. Of course, if power consumption is not an issue and power is available, we could scale up to edge solutions such as Raspberry PI in order to optimise the execution time. We discourage platforms with embedded GPUs in such energy-constrained contexts where workloads do not take advantage of all the computational resources provided.

5 Conclusion and future work

Edge computing is leading the HPC-AI intersection in the IoT arena. However, the computational gap between the two disciplines is still huge. It requires simpler algorithms and platforms to offer novel solutions to emerging applications to make them viable in this context, i.e. meeting both runtime and power consumption requirements. In this paper, we analyse a wide range of edge computing platforms in the context of TinyML. In particular, we use two lightweight artificial neural networks, namely MLP and CNN, for indoor temperature prediction in greenhouses. We use them as benchmarks to evaluate four different edge computing platforms; an Arduino microcontroller, a Raspberry-Pi, and two platforms of the familiar Nvidia Jetson.

Our results show that a balance between algorithmic complexity, the accuracy of the results obtained, and energy efficiency is essential to obtain robust and operational systems in real-world environments. We claim that the best hardware-software combination for the problem analysed would be the *CNN60m* model running on the Arduino microcontroller. We also observe the low impact of introducing more computation at this level, as seen in the energy efficiency numbers of Nvidia's Jetson family. Actually, the energy consumption and thus the carbon footprint, is one a critical factor in greenhouses, while the accuracy of the model is not so critical as long as the error is below a certain threshold. That's why MLP models running on the Arduino microcontroller may be a better choice, as they offer almost an order of magnitude lower power consumption. We also conclude that platforms with integrated GPUs in these energy-constrained contexts may not be sufficiently fruitful, unless the model run is sufficiently complex.

We recognise that the problem addressed in this paper is relatively simple. When scaled up to a more complex problem, e.g. multivariate greenhouse modelling or smart irrigation, more complex models will need to be developed to achieve adequate accuracy and require computational power. We envision this algorithmic complexity as a chance to also scale in computational horsepower at the edge.

Author Contributions Conceptualization was contributed by JMC and RME; methodology was contributed by JMC, RME and ABC.; software was contributed by JMG and JLP; validation was contributed by JMC, RME, PM and ABC; formal analysis was contributed by ABC, JMC, PM and RME; investigation was contributed by JMG and JMC; data curation was contributed by ABC, RME and JMG; writing—original draft preparation, was contributed by JMG, JLP, RME, JMC and AB; writing—review

and editing, was contributed by PM and JMC; visualisation was contributed by JLP, JMG and RME; supervision was contributed by JMC and PM; project administration was contributed by JMC; funding acquisition was contributed by JMC. All authors have read and agreed to the published version of the manuscript.

Availability of data and materials All data and materials are available on request from the authors of this paper.

Declarations

Conflict of interest The authors declare they do not have any conflict of interest.

Ethical approval Not applicable.

Funding This work is derived from R &D projects RTC2019-007159-5, as well as the Ramon y Cajal Grant RYC2018-025580-I, funded by MCIN/AEI/10.13039/501100011033, "FSE invest in your future" and "ERDF A way of making Europe".

References

1. Feki MA, Kawsar F, Boussard M, Trappeniers L (2013) The internet of things: the next technological revolution. Computer 46(2):24–25
2. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (iot): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
3. Tahsien SM, Karimipour H, Spachos P (2020) Machine learning based solutions for security of internet of things (iot): a survey. J Netw Comput Appl 161:102630
4. Papadokostaki K, Mastorakis G, Panagiotakis S, Mavromoustakis CX, Dobre, C, Batalla JM (2017) Handling big data in the era of internet of things (IoT). Springer
5. Satyanarayanan M (2017) The emergence of edge computing. Computer 50(1):30–39
6. Capra M, Peloso R, Masera G, Ruo Roch M, Martina M (2019) Edge computing: a survey on the hardware requirements in the internet of things world. Future Internet 11(4):100
7. Warden P, Situnayake D (2019) TinyML. O'Reilly Media, Incorporated
8. Portilla J, Mujica G, Lee J-S, Riesgo T (2019) The extreme edge at the bottom of the internet of things: a review. IEEE Sens J 19(9):3179–3190
9. Deng L, Yu D (2014) Deep learning: methods and applications. Found Trends Signal Process 7(3–4):197–387
10. Guillén-Navarro M, Martínez-España R, Bueno-Crespo A, Ayuso B, Moreno JL, Cecilia JM (2019) An LSTM deep learning scheme for prediction of low temperatures in agriculture. IOS Press, Amsterdam, pp 130–138
11. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press, Cambridge
12. Abhishek K, Singh M, Ghosh S, Anand A (2012) Weather forecasting model using artificial neural network. Procedia Technol 4:311–318
13. Lee S, Lee Y-S, Son Y (2020) Forecasting daily temperatures with different time interval data using deep neural networks. Appl Sci 10:1609
14. Zhang Z, Dong Y (2020) Temperature forecasting via convolutional recurrent neural networks based on time-series data. Complexity
15. Jung D-H, Kim HS, Jhin C, Kim H-J, Park SH (2020) Time-serial analysis of deep neural network models for prediction of climatic conditions inside a greenhouse. Comput Electron Agric 173:105402
16. Codeluppi G, Cilfone A, Davoli L, Ferrari G Ai at the edge: a smart gateway for greenhouse air temperature forecasting. In: 2020 IEEE international workshop on metrology for agriculture and forestry (MetroAgriFor), pp 348–353. IEEE
17. Guillén MA, Llanes A, Imbernón B, Martínez-España R, Bueno-Crespo A, Cano J-C, Cecilia JM (2021) Performance evaluation of edge-computing platforms for the prediction of low temperatures in agriculture using deep learning. J Supercomput 77(1):818–840

18. Codeluppi G, Davoli L, Ferrari G (2021) Forecasting air temperature on edge devices with embedded AI. Sensors 21(12):3973
19. Chang Z, Liu S, Xiong X, Cai Z, Tu G (2021) A survey of recent advances in edge-computing-powered artificial intelligence of things. IEEE Internet of Things J
20. Dubey AK, Kumar A, García-Díaz V, Sharma AK, Kanhaiya K (2021) Study and analysis of SARIMA and LSTM in forecasting time series data. Sustain Energy Technol Assess 47:101474
21. Seshadri K, Akin B, Laudon J, Narayanaswami R, Yazdanbakhsh A (2021) An evaluation of edge tpu accelerators for convolutional neural networks. arXiv preprint arXiv:2102.10423
22. Rashid N, Demirel BU, Al Faruque MA (2022) Ahar: Adaptive cnn for energy-efficient human activity recognition in low-power edge devices. IEEE Internet of Things J
23. Cruz M, Mafra S, Teixeira E, Figueiredo F (2022) Smart strawberry farming using edge computing and IOT. Sensors 22(15):5866
24. Feng B, Ding Z, Jiang C (2022) Fast: A forecasting model with adaptive sliding window and time locality integration for dynamic cloud workloads. IEEE Trans Serv Comput
25. Ding Z, Feng B, Jiang C (2022) Coin: a container workload prediction model focusing on common and individual changes in workloads. IEEE Trans Parallel Distrib Syst 33(12):4738–4751
26. Alongi F, Ghielmetti N, Pau D, Terraneo F, Fornaciari W (2020) Tiny neural networks for environmental predictions: an integrated approach with miosix. In: 2020 IEEE International Conference on Smart Computing (SMARTCOMP), pp 350–355. IEEE
27. Pettit A (1979) A non-parametric approach to the change-point problem. Appl Stat 28(2):126–135
28. Bishop CM et al (1995) Neural networks for pattern recognition. Oxford University Press, Oxford
29. Tadeusiewicz R (1995) Neural networks: a comprehensive foundation: by Simon HAYKIN; Macmillan College Publishing, New York, USA; IEEE Press, New York, USA; IEEE Computer Society Press, Los Alamitos, CA, USA; 1994; 696 pp 69–95; ISBN: 0-02-352761-7. Pergamon
30. Li Y, Hao Z, Lei H (2016) Survey of convolutional neural network. J Comput Appl 36(9):2508
31. Sahu M, Dash R (2021) A survey on deep learning: convolution neural network (CNN). Springer, Berlin
32. Tensorflow: Tensorflow lite for microcontrollers. https://www.tensorflow.org/lite/microcontrollers. Accessed 2021-07-06
33. Inc MT: PAC1934 USB C POWERMETER. https://www.microchip.com/en-us/development-tool/ADM00921

Authors and Affiliations

Juan Morales-García[1] · Andrés Bueno-Crespo[1] · Raquel Martínez-España[2] · Juan-Luis Posadas[3] · Pietro Manzoni[3] · José M. Cecilia[3]

Andrés Bueno-Crespo
abueno@ucam.edu

Raquel Martínez-España
raquel.m.e@um.es

Juan-Luis Posadas
jposadas@disca.upv.es

Pietro Manzoni
pmanzoni@disca.upv.es

José M. Cecilia
jmcecilia@disca.upv.es

[1] Computer Science Department, Catholic University of Murcia (UCAM), Av. de los Jerónimos, 135, 30107 Murcia, Murcia, Spain

[2] Information and Communications Engineering Department, University of Murcia (UM), C. Campus Universitario, 11, 30100 Murcia, Murcia, Spain

[3] Computer and Systems Informatics Department, Polytechnic University of Valencia (UPV), Camino de Vera, s/n, 46022 Valencia, Valencia, Spain

2.3. SEPARATE: A tightly coupled, seamless IoT infrastructure for deploying AI algorithms in smart agriculture environments

Los detalles de la tercera publicación del compendio se muestran en la tabla 3.

Detalles del artículo	
Título	SEPARATE: A tightly coupled, seamless IoT infrastructure for deploying AI algorithms in smart agriculture environments
Revista	Internet of Things
Autores	Juan Morales-García, Andrés Bueno-Crespo, Raquel Martínez-España, Francisco J. García, Sergio Ros, Julio Fernández-Pedauyé, José M. Cecilia
Año de publicación	2023
DOI	10.1016/j.iot.2023.100734
Estado	Publicado

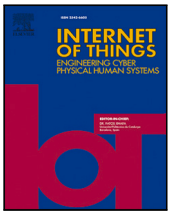
Tabla 3: Detalles del tercer artículo del compendio de publicaciones.

Internet of Things 22 (2023) 100734

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

SEPARATE: A tightly coupled, seamless IoT infrastructure for deploying AI algorithms in smart agriculture environments

Juan Morales-García [a,*], Andrés Bueno-Crespo [a], Raquel Martínez-España [b], Francisco J. García [c], Sergio Ros [c], Julio Fernández-Pedauyé [d], José M. Cecilia [d]

[a] Computer Science Department, Catholic University of Murcia (UCAM), Murcia, Spain
[b] Information and Communications Engineering Department, University of Murcia (UM), Murcia, Spain
[c] Nutricontrol, S.L., Cartagena, Spain
[d] Computer and Systems Informatics Department, Polytechnic University of Valencia (UPV), Valencia, Spain

ARTICLE INFO

Keywords:
Publish/Subscribe infrastructure
Edge computing
Machine learning
Deep learning
Internet of Things
Smart agriculture

ABSTRACT

Precision agriculture generates large datasets from IoT infrastructures deployed for continuous crop monitoring. This data requires analysis to usefully transform this data deluge into insights that can deliver value-generating services to farmers in a timely manner. This paper introduces *SEPARATE*; a dynamic interoperable and decentralized infrastructure for executing both, training and inference stages of deep learning (DL) algorithms in smart agriculture scenarios. The presented infrastructure allows the execution of the inference stage at the edge, achieving a highly efficient and responsive local temperature prediction service to take actions based on the predictions generated. Moreover, the training stage is offloaded to the cloud along with the generated historical data, allowing the trained model to be periodically updated at the edge. On the one hand, our results show that the Convolutional Neural Network model together with the Long Short-Term Memory technique (CNNLSTM) obtains the best results in both prediction accuracy and computational time. On the other hand, an analysis has been carried out to determine how often the model must be retrained, obtaining results that indicate that from day 9–10, it would be necessary to retrain the model, although, until day 20, the precision is not greatly reduced. Moreover, the *SEPARATE* infrastructure enables the execution of real-time inference from sensor-generated data and seamless model retraining in an operational greenhouse for temperature forecast with satisfactory performance.

1. Introduction

Precision agriculture is increasingly making use of new technologies in order to improve profits and reduce costs [1]. In a globalized world where costs are increasing day by day, it is important to have autonomous systems that allow farmers to better control their crops [2]. Monitoring and anticipation in decision-making can save a lot of production costs and crop losses [3]. Greenhouses, due to their structure, allow crop control, both of growth, pests, and climatic variables that directly influence crop yields [4]. For this control, the devices and sensors provided by the Internet of Things (IoT) provide great versatility and convenience for efficient data capture.

Current operational IoT solutions in the area of smart agriculture in real-world environments are mainly based on centralized IoT systems, where data is sent to a centralized cloud-based architecture, processed, and further analyzed [5]. However, agricultural

* Corresponding author.
E-mail address: jmorales8@ucam.edu (J. Morales-García).

https://doi.org/10.1016/j.iot.2023.100734
Received 25 November 2022; Received in revised form 20 February 2023; Accepted 21 February 2023
Available online 24 February 2023

environments are often located in rural areas where connectivity and power supply are limited [6]. Indeed, a dynamic, interoperable, and decentralized architecture is mandatory in these environments to achieve highly efficient and responsive data-based services. Some solutions have been proposed that bring the execution of machine learning (ML) and even deep learning (DL) algorithms close to the data capture, i.e. at the edge, in agricultural environments [7–9]. Edge computing [10] is becoming a widely used approach towards decentralization, where preliminary computations on data are carried out in (or close to) the data capture devices, providing a number of advantages, including energy savings, responsive application and service development, highly scalable, reliable and secure system design.

However, computational devices that are available at the edge typically rely on batteries or energy harvesters, leading to ultra-low power designs, but also limiting the workloads to be executed on them. Actually, edge Computing is limited to the inference stage of Machine Learning/Deep Learning (ML/DL) algorithms in what is recently called TinyML [11]. This makes sense because at this level of the IoT infrastructure, very little computational horsepower is available and thus computationally heavy workloads, such as those found at the training stage of ML/DL algorithm cannot be executed in a reasonable timeframe and with a manageable energy budget. However, training ML/DL algorithm periodically is mandatory to achieve more accurate results in these environments of rapidly changing conditions, so influenced by the abrupt changes brought about by climate change [12]. In this paper, we present *SEPARATE*; a seamless and tightly coupled IoT infrastructure for developing training and inference of ML/DL algorithms in operational smart agriculture environments. *SEPARATE* relies on a publish/subscribe (Pub/Sub) system based on Message Queuing Telemetry Transport (MQTT) protocol to perform data analytics at the edge of an IoT infrastructure deployed in an operational greenhouse located in Murcia (Spain). Particularly, the *SEPARATE* infrastructure receives information through MQTT from the air temperature sensors deployed inside the greenhouse that feed ML/DL models to predict the climatic state of the greenhouse in the following hours (i.e., inference stage). It is also important to study the need to retrain the prediction model to always have the best available precision without affecting response times. Moreover, *SEPARATE* also sends the data to the cloud via MQTT to store the historical data and thus to periodically retrain the model to be updated at the inference stage and thus provide a more accurate forecast. The main contributions of the paper include the following:

1. The *SEPARATE* infrastructure is introduced to provide an interoperable and decentralized dynamic architecture for ML/DL training and inference.
2. *SEPARATE* is designed for a real and operational IoT infrastructure. Our developments and tests have been carried out in an operational greenhouse deployed in Murcia (Spain)
3. Different ML/DL models are considered in terms of execution time and accuracy to establish the best combination in the quality and performance tradeoff.
4. Study and analysis of the days needed to retrain the best model, without significantly losing precision.
5. *SEPARATE* offers an infrastructure not only for monitoring but also for predicting actions in advance, thanks to its prediction models for several hours ahead.

The rest of the paper is organized as follows. Section 2 shows related works within the umbrella of ML/DL as applied to Pub/Sub solutions in forecasting climatic variables in greenhouses. Section 3 describes in detail the different approaches proposed and the artificial intelligence methods used to compare an evaluate results. Section 4 shows the performance results for the Pub/Sub solution and for the artificial intelligence models before discussing them in Section 5. Finally, Section 6 presents the main conclusions and discusses future works.

2. Related works

The MQTT communications protocol is increasingly being used in precision agriculture, replacing the HTTP protocol, [13]. Although the HTTP protocol uses a request/response architecture and HTTP can transfer a large number of data in small packets that can cause a large overhead. Therefore, due to this overhead, communication for IoT services via this protocol can cause serious bandwidth problems. Moreover, all HTTP calls are stateless, which leads to authentication every time you connect, as you connect to the IP or URL to make the REST API calls, the session is not saved and thus, after getting the response, the device closes the connection. This is due to possible network overload [14]. In contrast, MQTT (Message Queue Telemetry Transport) is a lightweight protocol that is designed for the IoT ecosystem. It was invented by IBM, is based on the OSI TCP/IP model, and has a very lightweight application layer with a header size of 2 bytes. MQTT follows an asymmetric architecture and operates under the publish and subscribe protocol. It is designed to send a message to one or more devices with low latency but is not recommended for sending large amounts of data. Given its characteristics, it is very useful for use in IoT systems, since among its advantages we find the secure delivery of the message, thus avoiding the loss of data [15,16]. Given the advantages of the MQTT protocol, it is increasingly used in IoT environments and in the world of precision agriculture. For instance, in [17] the authors propose a precision agriculture system based on wireless sensor networks with the MQTT protocol for monitoring and controlling the environmental conditions of a greenhouse by collecting information on humidity, temperature, light, and nutritional needs of the plants. In [18], the design and implementation of an intelligent irrigation system to automate the irrigation system in agricultural fields are proposed. The system is a near real-time system using the MQTT protocol for communications between the sensor side and the client side. The authors of [2] present an open-source platform covering automated precision agriculture scenarios. For this purpose, the platform is distributed on three levels, Cyber–Physical Systems, Edge computing, and Cloud. To connect the sensors and actuators with the end user, they use IoT technologies and protocols such as MQTT, the Constrained Application Protocol (CoAP), and FIWARE, among others.

Fig. 1. Targeted operational greenhouse located at Murcia (Spain).

The platform is successfully tested in a hydroponic greenhouse. In [19], authors proposed the design of a remote monitoring and control system for greenhouses. The system uses IoT to collect greenhouse parameters such as temperature, relative humidity, and luminosity. After collecting the data and sending it to the server, the IoT polls the data on the internet through the MQTT protocol and compares it with existing parameters in order to send a correction to the greenhouse if necessary. A system for monitoring and controlling microclimatic parameters of a greenhouse is proposed in [20]. In this case, the system collects temperature and humidity values as well as gas values. The data is sent through an Arduino to a raspberry via the MQTT protocol.

Another aspect of agriculture where IoT systems and the MQTT protocol is also very useful is for the prevention and care of bees to avoid Colony Collapse Disorder, a disorder still under study. To obtain data for this study, in [21] the authors propose the design of a near real-time system that collects temperature, humidity, and weight data and sends it via MQTT to a ThingsBoard for analysis, the system called Beemon works continuously and in open-air hives.

As can be seen, the proposed systems are used to monitor and control different agricultural systems, from different points of view. The difference with our proposal is that in our case, in addition to carrying out monitoring tasks, we also carry out advance prediction tasks, in order to take measures before any event occurs, thus achieving a better optimization of resources.

3. Materials and methods

This section provides the materials and methods of the *SEPARATE* infrastructure. As previously mentioned, it provides an interoperable and decentralized dynamic architecture for ML/DL training and inference in an operational greenhouse. Therefore, we introduce the operational greenhouse where *SEPARATE* is deployed before providing to the reader the main insights of *SEPARATE* infrastructure. The case study here presented aims at forecasting the internal temperature of this greenhouse through different ML/DL methods that are also presented, showing the main parameters used for the evaluation carried out in the next section.

3.1. Operational greenhouse

Fig. 1 shows the operational greenhouse targeted for this study, namely *ETIFA*. *ETIFA* is an operational greenhouse hosted by NUTRICONTROL; a Spanish leading company in developing automatic fertigation and Climate control technology. *ETIFA* is located in Murcia (South-eastern Spain), a semi-arid region where the average annual temperature is around 25 °C. This greenhouse has a surface area of 50 m^2 and operates with a system for climate control and fertigation.

ETIFA fertigation and climate control is carried out by NUTRICONTROL's OPTIMUM system; a complete solution for climate management and fertigation of these environments. The NUTRICONTROL's OPTIMUM system is orchestrated by a CPU-based node (OPTIMUM Orchestator, from now on) where all sensors (input/output) are plugged into in a modular way. Among the sensors available in *ETIFA* are temperature, humidity, radiation, and wind speed, just to name a few.

Of particular interest to us is the air temperature inside the greenhouse as it is the target for the forecast carried out in this paper. This variable is measured every 5 min in the greenhouse, providing near-real-time (NRT) continuous measurements to take actions that can increase/decrease the greenhouse temperature to reach the ideal temperature of the crop being grown.

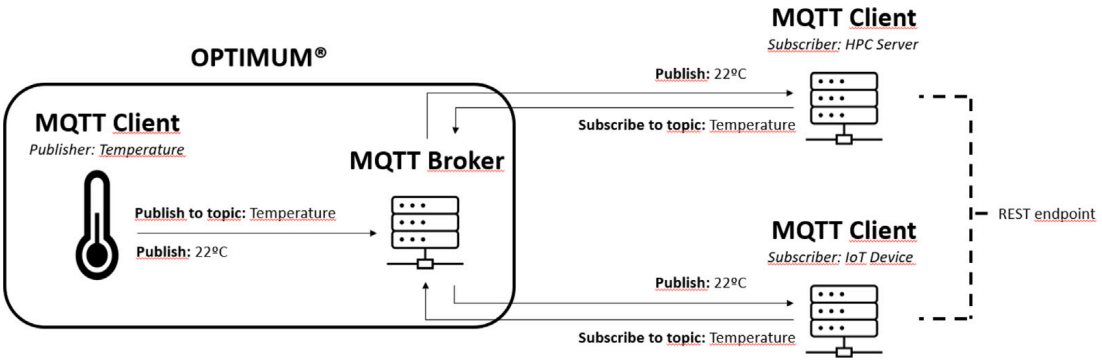


Fig. 2. *SEPARATE* main building blocks based on MQTT schema.

3.2. SEPARATE infrastructure

The *SEPARATE* infrastructure relies on the MQTT standard protocol [22]. MQTT is a lightweight Pub/Sub messaging transport system where sensors can publish the generated data (e.g. temperature sensor) and several nodes can be subscribed to receive values in NRT. Fig. 2 shows the main *SEPARATE* building blocks that are deployed in the greenhouse. First, the MQTT broker or server, deployed in the OPTIMUM orchestrator, is responsible for dispatching messages between the sender (or publisher) and the appropriate receivers. *SEPARATE* is based on Java-based open source HIVEMQ Community Edition (CE).[1]

MQTT clients are subscribed to and publish information on a particular topic. In our case, the MQTT client is developed using the Eclipse Paho MQTT Python client library [23] that enables applications to connect to an MQTT broker to publish messages and to subscribe to topics and receive published messages. *SEPARATE* infrastructure is based on three MQTT clients. The first client is hosted in the OPTIMUM orchestrator and publishes data to the topic "Temperature" from the greenhouse as soon as it is received from the sensors. The second and third clients are subscribed to the same topic but are deployed in different computing nodes, depending on the task associated with each of them.

The second MQTT client is deployed on an Nvidia Jetson Nano in the greenhouse; very close to the capture node, i.e. at the edge. This client is designed to run the ML/DL inference of the model to forecast the indoor temperature of the greenhouse for the next few hours. In this way, the MQTT client periodically receives the data temperature that is stored in a CSV file in the Jetson Nano. Another background process (i.e. cron job) runs the model inference every set time frame (e.g., every 15 min), which collects the information gathered by the MQTT client and generates the prediction. It is important to note that each time frame, the ML/DL inference is executed taking into account the last updated data generated by the greenhouse and thus more accurate predictions should be generated using these new records.

The last MQTT client is deployed in a cloud server hosted in our lab at UPV. This MQTT client is also subscribed to "Temperature" topic and once data is received from this topic, this cloud-based MQTT client stores it in an InfluxDB database where historical data is increasingly generated. On this cloud-based server, data quality controls, sanitization processes, etc. are carried out to prepare the data for an efficient training procedure. It is important to note that the training procedure does not need to be carried out on a sub-daily basis or even a daily basis. This is indeed a configuration parameter that is set according to the computational and accuracy tradeoff.

In preliminary evaluations conducted in the particular context of the ETIFA greenhouse, it has been empirically observed that trained models give predictions at the inference stage within the same quality range as models trained with data from up to a month before. However, important differences in the forecast accuracy start to become relevant if ML/DL models are trained with data of more than a month before the horizon is predicted. Depending on the application, the user may prefer a more conservative scenario and retrain more frequently to obtain the most accurate results, or be more computationally efficient and only train when significant differences in the results are really noticeable. It is worth noting that training the heavier ML/DL ETIFA models can take several computational days, as it will be shown in Section 4.2.

In any case, the model will be trained on the cloud server, and once trained, *SEPARATE* has to provide a mechanism to update the model trained on the edge in a transparent way to the user. The chosen way has been to enable a REST endpoint where a process on the edge node can update the model periodically. In our case, a month has been set by default although this can be easily changed in the configuration schema.

3.3. Datasets

For the evaluation of the accuracy of the AI models, two different representative points of the year have been taken into account; i.e. winter and summer. This dichotomy is due to the fact that these are the two points at which the climatological variables (and

[1] https://github.com/hivemq/hivemq-community-edition

Table 1
Dataset description.

Datasets	Start date	End date	# Instances
CLEAN-DS-15-SUMMER	18-12-18	06-06-21	86544
CLEAN-DS-15-WINTER	18-12-18	17-01-21	73104
CLEAN-DS-30-SUMMER	18-12-18	06-06-21	43273
CLEAN-DS-30-WINTER	18-12-18	17-01-21	36553
CLEAN-DS-60-SUMMER	18-12-18	06-06-21	21637
CLEAN-DS-60-WINTER	18-12-18	17-01-21	18277
DIRTY-DS-15-SUMMER	18-12-18	06-06-21	86544
DIRTY-DS-15-WINTER	18-12-18	17-01-21	73104
DIRTY-DS-30-SUMMER	18-12-18	06-06-21	43273
DIRTY-DS-30-WINTER	18-12-18	17-01-21	36553
DIRTY-DS-60-SUMMER	18-12-18	06-06-21	21637
DIRTY-DS-60-WINTER	18-12-18	17-01-21	18277
SMOOTH-DS-15-SUMMER	18-12-18	06-06-21	86544
SMOOTH-DS-15-WINTER	18-12-18	17-01-21	73104
SMOOTH-DS-30-SUMMER	18-12-18	06-06-21	43273
SMOOTH-DS-30-WINTER	18-12-18	17-01-21	36553
SMOOTH-DS-60-SUMMER	18-12-18	06-06-21	21637
SMOOTH-DS-60-WINTER	18-12-18	17-01-21	18277

their behavior) differ the most. Moreover, data time granularity has been also considered for the evaluation, i.e., the data are grouped into 15 min, 30 min, and 60 min periods. In this way, we can verify that (1) the model fits the data correctly, (2) it is able to predict at any point of the year and (3) that different time granularity can be applied depending on the needs of the problem. Finally, short-term (12 h) and long-term (24 h) prediction has also been taken into account.

Table 1 summarizes the description of the datasets that have been used to carry out the temperature prediction. It shows the start date of the data, the end date, and the number of instances contained in each dataset. Each dataset contains the temperature values of a greenhouse between the indicated dates, distinguishing whether the dataset ends on a summer or winter day. It is important to clarify that in southeastern Spain, June temperatures are already summer temperatures.

3.4. Artificial Intelligence models

This section introduces four ML/DL models that have been used for the evaluation of *SEPARATE*. We refer the reader to [24] for insights.

- **Artificial Neural Networks (ANN)**: A neural network is a model that mimics the way a set of biological neurons works. Although its use in classification is more widespread, it can also be used in regression models. A single perceptron (or artificial neuron) can be imagined as a logistic regression. The artificial neural network, or ANN, is a group of multiple perceptrons in each layer forming a multilayer perceptron (MLP). The MLP we use in this work is composed of three layers: input, hidden, and output. The input layer receives the input features, the hidden layer processes the inputs and the output layer produces the output. Essentially, each layer tries to learn certain weights. Artificial neural networks have the ability to learn any complex relationship between input and output because they use an activation function that allows them to learn non-linear properties in the network. [25,26].
- **Convolutional Neural Network (CNN)**: Convolutional neural network models are used in different applications and domains, where they are most frequently used in imaging for classification. However, they are also used in regression, where they can be used using time series by transforming the data to adapt them to the inputs of the convolutional network. A CNN is made up of blocks of filters, which, through convolution operations, allow the relevant features to be extracted from the input. One of the advantages of CNNs over conventional neural networks (ANNs) is the automatic learning of the filters so that the necessary and most relevant features are obtained from the input data. [27].
- **Long Short-Term Memory (LSTM)**: this model of DL is commonly used because in addition to working with time series like recurrent models, but with the advantage that this network allows for long-term memory. LSTM is a type of recurrent neural architecture with a state memory and multilayer cell structure [28]. LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate (Fig. 3, only the LSTM layer). The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. The LSTM differs from a classic recurrent network in that it does not overwrite its content at each time step but is able to decide whether to keep the existing memory through the introduced doors. If the LSTM unit detects an important characteristic of an input sequence at an early stage, it carries this information over long distances, therefore it detects long-distance dependencies.
- **Convolutional Neural Network + Long Short-Term Memory (CNNLSTM)**: This model is a combination of CNN and LSTM (also known as ConvLSTM). It presents a convolutional network where the MLP layer fully connected to the final layer has been replaced by an LSTM network. Therefore, the CNN is in charge of automatically extracting the input features and the LSTM is in charge of obtaining the regression results (see Fig. 3).

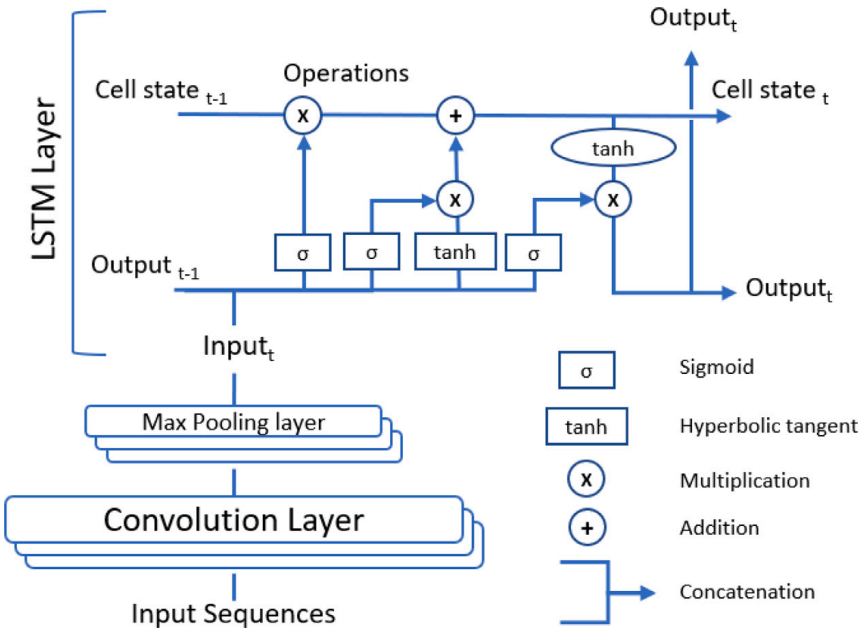


Fig. 3. Model CNNLSTM. In the first stage, the convolutional layer is in charge of extracting the feature vector so that finally the LSTM layer performs the forecast.

Table 2
Hyperparameters used for each model. A dash in a cell indicates that the model does not have that parameter.

Hyperparameter	MLP	CNN	LSTM	CNNLSTM
Units	70	–	70	–
Filters	–	64	–	64
Kernel size	–	1	–	2
Strides	–	1	–	4
Activation function	Tanh	Tanh	Tanh	Tanh
Batch size	2880	2880	2880	2880
Epochs (+ *EarlyStopping*)	3000	3000	3000	3000
Optimizer	Adam	Adam	Adam	Adam
Loss function	MSE	MSE	MSE	MSE
Learning rate (+ *ReduceLROnPlateau*)	0.003	0.003	0.003	0.003

Table 2 shows all hyperparameters (rows) used for each model (columns). A brief description of the meaning of each hyperparameter is given below:

- **Units**: Number of neurons used in hidden layers.
- **Filters**: Features detector.
- **Kernel size**: Filters matrix used to extract the features from the dataset.
- **Strides**: Number of pixels shifts over the input matrix.
- **Activation function**: Function that decides if a neuron should be (or not) activated.
- **Batch size**: Size of bach used for training/forecasting.
- **Epochs**: Number of epoch used in training.
- **Optimizer**: Function that optimizes the learning of an artificial intelligence model, updating its neurons' weights depending on the error evaluation.
- **Loss function**: Function used to evaluate the error of the model in each epoch.
- **Learning rate**: Percentage change with which weights are updated at each iteration.

4. Results

This section shows the results of the different experiments that have been carried out. They can be divided into two main sets. On one side, an experiment is proposed to evaluate the effectiveness of the NRT system based on *SEPARATE* (see Section 4.1). While, on the other side, an experiment will be proposed to evaluate the accuracy of ML/DL models in an NRT environment (see

Table 3
Execution time in seconds obtained by running the MLP model on CPU and GPU included in the HPC server platform. Execution time of training and inference (12 h and 24 h) are provided.

Platform	CPU			GPU		
Times	Training	Inference		Training	Inference	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	713.861	2.276	4.472	500.245	2.470	4.862
CLEAN-DS-15-WINTER	628.725	2.246	4.737	445.597	2.428	5.022
CLEAN-DS-30-SUMMER	390.665	1.169	2.266	311.396	1.246	2.419
CLEAN-DS-30-WINTER	358.576	1.165	2.253	293.535	1.252	2.410
CLEAN-DS-60-SUMMER	275.897	0.604	1.151	248.403	0.629	1.194
CLEAN-DS-60-WINTER	264.678	0.596	1.113	241.099	0.637	1.193
DIRTY-DS-15-SUMMER	699.936	2.290	4.514	494.385	2.425	4.900
DIRTY-DS-15-WINTER	645.416	2.374	4.615	440.965	2.472	4.828
DIRTY-DS-30-SUMMER	388.948	1.164	2.243	311.467	1.248	2.406
DIRTY-DS-30-WINTER	354.699	1.159	2.250	294.019	1.226	2.388
DIRTY-DS-60-SUMMER	276.120	0.596	1.146	248.354	0.644	1.197
DIRTY-DS-60-WINTER	264.019	0.591	1.138	242.394	0.648	1.207
SMOOTH-DS-15-SUMMER	729.050	2.490	4.512	491.324	2.639	4.817
SMOOTH-DS-15-WINTER	633.765	2.321	4.570	452.423	2.403	4.809
SMOOTH-DS-30-SUMMER	392.934	1.155	2.594	308.026	1.240	2.685
SMOOTH-DS-30-WINTER	358.637	1.155	2.254	293.757	1.235	2.395
SMOOTH-DS-60-SUMMER	274.794	0.595	1.124	247.762	0.627	1.205
SMOOTH-DS-60-WINTER	263.945	0.600	1.152	242.576	0.642	1.222

Section 4.2). In addition, a study is carried out on the need to retrain the model from time to time so as not to lose precision in the predictions. These results are broadly discussed in Section 5.

4.1. Pub/Sub solution evaluation

In the following, the execution time on HPC and edge computing platforms for training and inference of ML/DL techniques is shown and analyzed individually. This experiment leads us to argue the computational differences between the two devices and the need for *SEPARATE* to coordinate training and inference in this context.

4.1.1. HPC evaluation

The HPC computing node is evaluated by running the training and inference for the different ML/DL models described in Section 3.4. This node is composed of an AMD EPYC 7282, 64 CPUs (2 sockets x 16 cores per socket x 2 threads per core) at 2.8 MHz for each core. Moreover, it also contains two NVIDIA A100-PCIE-40 GB (Tesla architecture) and it is endowed with up to 256 GB and 2 TB SSD.

Tables 3, 4, 5, 6 show the MLP, CNN, LSTM and CNNLSTM execution time on the HPC server targeted for the multicore CPU or a GPU, respectively. Each row of the table represents the performance times for each of the datasets described in Section 3.3. Each column of the table represents the performance times studied and is grouped into training and inference, for each of the two platforms studied (CPU and GPU). For inference, a distinction is made between 12 h and 24 h inference.

In particular, Table 3 shows that the MLP training time on CPU range from 263.945 to 729.050 s, while on GPU they range from 241.099 to 500.245 s, obtaining up to 1.5X performance in the best scenario. Inference times are almost identical on both CPU and GPU, being a few milliseconds slower on GPU, and ranging between 0.592 and 5.022 s. Regarding the inference between 12 and 24 h, there is practically no difference, since the 24 h time is practically twice as long as the 12 h time, which is consistent since twice as many values are inferred.

Table 4 shows that the CNN training time on CPU range from 2,089.193 to 33,877.128 s, while on GPU they range from 275.392 to 755.894 s, obtaining up to 10X performance in the best-case scenario. Inference times are almost identical on both CPU and GPU, being a few milliseconds slower on GPU, and ranging between 0.571 and 5.166 s.

Table 5 shows that the LSTM training time on CPU range from 7,821.983 to 148,153.696 s, while on GPU they range from 446.884 to 3,419.390 s, obtaining a 15X performance. Inference times are almost identical on both CPU and GPU, being a few milliseconds slower on CPU, and ranging between 1.003 and 6.020 s.

Finally, Table 6 shows that the CNNLSTM training time on CPU range from 1,831.9153 to 31,471.342, while on GPU they range from 323.845 to 890.787, obtaining up to 20X performance. Inference times are almost identical on both CPU and GPU, being a few milliseconds slower on GPU, and ranging between 0.969 and 5.049.

In conclusion for the 4 models run on the server, we have that the GPU execution for training is much faster between 1.5 and 20x at best. The speed difference between CPU and GPU increases when the model is more complex, the complexity hierarchy being MLP, CNN, LSTM, and CNNLSTM models. In the case of inference, the same situation occurs for all four models, i.e., execution on the CPU is slightly faster than on the GPU. This has a logical explanation since in the case of inference, no large computational power is needed.

Table 4
Execution time in seconds obtained by running the CNN model on CPU and GPU included in the HPC server platform. Execution time of training and inference (12 h and 24 h) are provided.

Platform	CPU			GPU		
Times	Training	Inference		Training	Inference	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	33807.923	2.316	4.507	749.809	2.436	4.839
CLEAN-DS-15-WINTER	27835.258	2.348	4.509	660.638	2.463	4.822
CLEAN-DS-30-SUMMER	8460.801	1.192	2.518	404.968	1.236	2.716
CLEAN-DS-30-WINTER	7237.790	1.175	2.223	384.064	1.249	2.384
CLEAN-DS-60-SUMMER	2454.417	0.616	1.122	298.052	0.684	1.232
CLEAN-DS-60-WINTER	2119.112	0.580	1.121	286.328	0.658	1.210
DIRTY-DS-15-SUMMER	33613.773	2.309	4.503	755.894	2.434	4.778
DIRTY-DS-15-WINTER	28009.784	2.405	4.575	667.642	6.271	4.811
DIRTY-DS-30-SUMMER	8423.981	1.189	2.234	408.309	1.241	2.376
DIRTY-DS-30-WINTER	7239.152	1.192	2.246	399.793	6.606	2.426
DIRTY-DS-60-SUMMER	2463.910	0.628	1.145	299.358	0.658	1.221
DIRTY-DS-60-WINTER	2089.193	0.607	1.110	299.714	6.090	1.211
SMOOTH-DS-15-SUMMER	33877.128	2.323	4.477	741.246	2.475	4.776
SMOOTH-DS-15-WINTER	27866.877	2.318	4.943	666.987	2.503	5.166
SMOOTH-DS-30-SUMMER	8456.179	1.165	2.284	400.301	1.265	2.383
SMOOTH-DS-30-WINTER	7228.456	1.454	2.317	377.420	1.669	2.397
SMOOTH-DS-60-SUMMER	2456.905	0.571	1.140	289.699	0.661	1.203

Table 5
Execution time in seconds obtained by running the LSTM model on CPU and GPU included in the HPC server platform. Execution time of training and inference (12 h and 24 h) are provided.

Platform	CPU			GPU		
Times	Training	Inference		Training	Inference	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	147642.328	3.290	5.605	3362.940	2.966	5.262
CLEAN-DS-15-WINTER	125425.139	3.363	5.973	2931.164	2.974	5.536
CLEAN-DS-30-SUMMER	37469.883	1.751	2.643	1136.169	1.634	2.467
CLEAN-DS-30-WINTER	31228.075	1.723	2.963	1001.663	1.633	2.887
CLEAN-DS-60-SUMMER	8983.771	1.003	1.159	481.747	1.013	1.219
CLEAN-DS-60-WINTER	7821.983	1.015	1.194	446.884	1.015	1.216
DIRTY-DS-15-SUMMER	147086.625	3.582	5.778	3419.389	3.284	5.173
DIRTY-DS-15-WINTER	125981.805	3.363	5.688	2988.742	3.048	5.275
DIRTY-DS-30-SUMMER	37478.132	1.706	2.496	1132.158	1.636	2.488
DIRTY-DS-30-WINTER	31363.002	1.721	2.576	998.420	1.639	2.460
DIRTY-DS-60-SUMMER	8994.831	1.025	1.581	479.596	1.018	1.749
DIRTY-DS-60-WINTER	7892.433	1.412	1.211	446.892	1.431	1.176
SMOOTH-DS-15-SUMMER	148153.696	3.290	5.654	3414.785	3.018	5.227
SMOOTH-DS-15-WINTER	126536.579	3.307	6.020	2991.940	3.291	5.263
SMOOTH-DS-30-SUMMER	37245.633	1.729	2.482	1122.942	1.630	2.467
SMOOTH-DS-30-WINTER	31630.357	1.705	2.617	1003.449	1.633	2.486
SMOOTH-DS-60-SUMMER	8994.010	1.014	1.189	482.858	1.007	1.227
SMOOTH-DS-60-WINTER	7841.688	1.020	1.195	448.608	1.029	1.217

4.1.2. Edge evaluation

The edge computing platform is evaluated by only running the inference for the different ML/DL models described in Section 3.4. In *SEPARATE*, only ML/DL inference is executed at this level of the infrastructure, as training is computationally forbidden. It is worth highlighting that the edge computing node is an Nvidia Jetson Nano that has a 5-core ARM Cortex-A57 MPCore CPU, 2MB L2, 128-core Maxwell GPU, and 4 GB 64-Bit LPDDR4 running at 25.6 GB/sec. Therefore, the CPU and GPU runs are analyzed to find out whether the inclusion of GPUs brings better performance results to inference in this scenario. Moreover, *SEPARATE* will send the ML/DL model into the edge device to proceed with the inference with the more updated model. Therefore, the execution time of loading an ML/DL model in the edge device once it is received is also reported below.

Tables 7, 8, 9, 10 show the MLP, CNN, LSTM and CNNLSTM execution time at the edge of both, the Load and Inference stages of the *SEPARATE* pipeline, respectively. Each row of the table shows the performance times for each of the datasets described in Section 3.3. Each column shows the execution times for these stages, for each of the two processors studied; i.e. CPU and GPU.

In particular, Table 7 shows that the MLP load time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 0.151 and 5.731 s. Inference time reported is also almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 2.351 and 20.356 s.

Table 6
Execution time in seconds obtained by running the CNNLSTM model on CPU and GPU included in the HPC server platform. Execution time of training and inference (12 h and 24 h) are provided.

Platform	CPU			GPU		
Times	Training	Inference		Training	Inference	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	31455.494	2.759	4.599	871.329	2.872	4.850
CLEAN-DS-15-WINTER	26227.227	3.148	4.683	775.974	3.189	4.841
CLEAN-DS-30-SUMMER	7883.942	1.588	2.300	487.813	1.647	2.420
CLEAN-DS-30-WINTER	6752.098	1.589	2.707	447.680	1.656	2.877
CLEAN-DS-60-SUMMER	2222.535	1.000	1.616	350.630	1.048	1.777
CLEAN-DS-60-WINTER	1920.816	1.009	1.125	334.212	1.039	1.212
DIRTY-DS-15-SUMMER	31329.182	2.778	4.611	868.439	2.871	4.855
DIRTY-DS-15-WINTER	26049.698	2.797	4.798	890.787	6.681	5.049
DIRTY-DS-30-SUMMER	7960.443	1.555	2.249	484.462	1.635	2.418
DIRTY-DS-30-WINTER	6788.469	1.606	2.245	467.469	6.823	2.415
DIRTY-DS-60-SUMMER	2221.137	0.969	1.060	351.492	1.041	1.203
DIRTY-DS-60-WINTER	1929.761	0.973	1.133	346.623	6.078	1.209
SMOOTH-DS-15-SUMMER	31471.342	2.731	4.541	852.136	2.853	4.848
SMOOTH-DS-15-WINTER	26336.821	2.713	4.564	767.090	2.843	4.803
SMOOTH-DS-30-SUMMER	7924.806	1.596	2.251	476.830	1.656	2.404
SMOOTH-DS-30-WINTER	6765.339	1.584	2.259	441.828	1.651	2.420
SMOOTH-DS-60-SUMMER	2222.474	1.011	1.147	343.871	1.030	1.199
SMOOTH-DS-60-WINTER	1831.915	0.999	1.142	323.847	1.042	1.202

Table 7
Execution time in seconds obtained by running the MLP model on CPU and GPU included in the edge computing platform. Times for a load of the ML model (Load) and execute the inference for the next 12 and 24 h (Inference) are provided.

Platform	CPU			GPU		
Times	Load	Inference		Load	Inference	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	0.154	9.364	19.212	0.199	9.912	20.136
CLEAN-DS-15-WINTER	0.155	9.192	18.792	0.275	10.069	20.088
CLEAN-DS-30-SUMMER	0.175	5.343	8.938	0.204	5.549	9.851
CLEAN-DS-30-WINTER	0.161	4.851	9.566	0.203	5.208	10.657
CLEAN-DS-60-SUMMER	0.192	2.817	4.787	0.251	3.124	4.839
CLEAN-DS-60-WINTER	0.167	2.326	4.525	0.270	2.684	4.902
DIRTY-DS-15-SUMMER	0.158	9.546	18.579	0.229	9.904	20.016
DIRTY-DS-15-WINTER	0.346	9.774	18.604	5.731	12.104	19.646
DIRTY-DS-30-SUMMER	0.151	4.655	9.667	0.234	4.383	10.310
DIRTY-DS-30-WINTER	0.153	4.469	8.719	0.244	5.389	9.983
DIRTY-DS-60-SUMMER	0.171	2.351	4.687	0.244	2.570	4.972
DIRTY-DS-60-WINTER	0.158	2.752	4.458	0.295	3.159	4.573
SMOOTH-DS-15-SUMMER	0.154	8.975	18.901	0.235	10.259	19.969
SMOOTH-DS-15-WINTER	0.156	8.774	18.566	0.211	9.967	20.356
SMOOTH-DS-30-SUMMER	0.153	4.478	9.508	0.242	5.056	10.472
SMOOTH-DS-30-WINTER	0.154	4.924	9.586	0.490	5.513	9.406
SMOOTH-DS-60-SUMMER	0.180	2.388	5.244	0.238	2.604	5.339
SMOOTH-DS-60-WINTER	0.165	2.357	4.788	0.360	2.757	4.674

Table 8 shows that the CNN load time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 0.273 and 17.968 s. Inference time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 2.320 and 21.107 s.

Table 9 shows that the LSTM load time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 1.457 and 8.081 s. Inference time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 3.793 and 25.917 s.

Finally, Table 10 shows that the CNNLSTM load time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 1.648 and 7.049 s. Inference time is almost identical on both CPU and GPU, being a few seconds slower on GPU, and ranging between 3.828 and 22.789 s.

In conclusion, execution on the edge is slightly faster on the CPU than on the GPU for all 4 models tested. This result is consistent with the results shown above for the execution on the server. As inference is an inexpensive process, the computational power offered by the GPU is not necessary. Evidently, inference on the Edge is somewhat slower than on the server, but with completely acceptable times.

Table 8
Execution time in seconds obtained by running the CNN model on CPU and GPU included in the edge computing platform. Times for load of the DL model (Load) and execute the inference for the next 12 and 24 h (Inference) are provided.

Platform	CPU			GPU		
Times	Load	Inference		Load	Inference	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	0.325	8.800	18.192	0.478	10.054	20.270
CLEAN-DS-15-WINTER	0.342	9.286	17.946	0.755	11.147	21.107
CLEAN-DS-30-SUMMER	0.273	4.806	9.560	0.405	5.257	10.004
CLEAN-DS-30-WINTER	0.278	4.509	8.893	0.417	5.002	9.567
CLEAN-DS-60-SUMMER	0.315	2.490	4.335	0.477	2.765	4.895
CLEAN-DS-60-WINTER	0.360	2.320	4.559	0.605	2.762	4.682
DIRTY-DS-15-SUMMER	0.403	9.511	18.579	0.569	10.436	20.527
DIRTY-DS-15-WINTER	0.739	9.745	18.280	9.830	28.747	20.656
DIRTY-DS-30-SUMMER	0.291	4.667	8.889	0.449	5.021	9.796
DIRTY-DS-30-WINTER	0.298	4.996	8.801	0.403	6.086	10.305
DIRTY-DS-60-SUMMER	0.269	2.843	4.683	0.574	3.293	4.727
DIRTY-DS-60-WINTER	0.364	2.555	4.509	0.656	3.234	4.935
SMOOTH-DS-15-SUMMER	0.325	9.843	17.758	17.968	10.806	20.851
SMOOTH-DS-15-WINTER	0.329	9.231	18.090	0.501	10.118	20.274
SMOOTH-DS-30-SUMMER	0.280	4.710	8.680	0.415	4.844	9.917
SMOOTH-DS-30-WINTER	0.320	5.105	8.919	0.400	5.556	9.913
SMOOTH-DS-60-SUMMER	0.367	2.439	4.605	0.424	2.740	4.653
SMOOTH-DS-60-WINTER	0.328	2.397	4.514	0.459	2.710	4.869

Table 9
Execution time in seconds obtained by running the LSTM model on CPU and GPU included in the edge computing platform. Times for load of the DL model (Load) and execute the inference for the next 12 and 24 h (Inference) are provided.

Platform	CPU			GPU		
Times	Load time	Forecast time		Load time	Forecast time	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	2.053	13.656	25.046	2.073	12.615	22.895
CLEAN-DS-15-WINTER	1.480	14.284	25.917	1.633	13.007	22.943
CLEAN-DS-30-SUMMER	2.294	6.581	10.545	2.283	7.006	11.355
CLEAN-DS-30-WINTER	1.504	6.802	11.056	1.569	7.157	10.739
CLEAN-DS-60-SUMMER	1.618	4.872	4.599	1.570	4.966	5.222
CLEAN-DS-60-WINTER	2.454	3.793	4.880	1.613	5.176	5.203
DIRTY-DS-15-SUMMER	1.502	14.437	24.876	1.534	13.426	21.957
DIRTY-DS-15-WINTER	2.070	14.355	24.823	8.081	25.461	22.661
DIRTY-DS-30-SUMMER	1.506	7.482	10.003	1.573	8.090	11.177
DIRTY-DS-30-WINTER	2.077	7.024	10.135	1.515	7.820	10.758
DIRTY-DS-60-SUMMER	1.477	4.200	4.978	1.540	4.272	5.305
DIRTY-DS-60-WINTER	1.457	3.919	4.540	1.566	4.518	5.583
SMOOTH-DS-15-SUMMER	1.559	13.906	24.823	1.550	13.544	21.932
SMOOTH-DS-15-WINTER	1.555	14.355	24.377	1.637	13.624	22.311
SMOOTH-DS-30-SUMMER	1.507	6.663	10.268	1.558	7.152	10.994
SMOOTH-DS-30-WINTER	1.548	7.959	10.392	1.550	7.270	12.191
SMOOTH-DS-60-SUMMER	1.534	4.005	5.018	1.528	4.376	5.483
SMOOTH-DS-60-WINTER	1.512	3.995	5.954	1.549	4.337	5.452

4.2. ML/DL models quality assessment

This section shows the results obtained when all ML/DL techniques are used to forecast the internal temperature of the greenhouse. Several metrics are provided to analyze the goodness of fit of the models. They are the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) which shows the gap between the forecast and actual values. MAE and RMSE are calculated by averaging the absolute difference between the predicted and actual values. Moreover, we also provide the R^2 that shows the variance of the forecast variable that is predictable from the actual variable. These metrics are shown for all ML/DL models, temporal granularity, and different evaluation periods (i.e. SUMMER and WINTER). It is important to note that these values were obtained using the Python-based Scikit-Learn library.

Table 11 shows the main quality figures achieved by the MLP model. This technique obtains the best result in MAE and RMSE with the CLEAN-DS-15-WINTER dataset for both 12 h and 24 h forecasts. The lowest MAE and RMSE is obtained with 12 h prediction with 1.276 °C and 1.538 °C respectively. The difference with the 24 h prediction is very small for all the datasets evaluated in general.

Table 12 reports the quality figures for the CNN technique. As with MLP, the results for the 12 h and 24 h forecasts are quite similar. However, the best result is obtained with the DIRTY-DS-30-WINTER dataset with a MAE and RMSE of 1.252 °C and 1.492 °C.

Table 10
Execution time in seconds obtained by running the CNNLSTM model on CPU and GPU included in the edge computing platform. Times for a load of the DL model (Load) and execute the inference for the next 12 and 24 h (Inference) are provided.

Platform	CPU			GPU		
Times	Load time	Forecast time		Load time	Forecast time	
Dataset		12 h	24 h		12 h	24 h
CLEAN-DS-15-SUMMER	1.751	11.511	19.149	2.138	13.424	22.707
CLEAN-DS-15-WINTER	1.914	12.009	19.610	3.977	13.725	22.218
CLEAN-DS-30-SUMMER	1.669	6.858	9.166	1.964	7.403	11.114
CLEAN-DS-30-WINTER	1.648	6.579	9.989	1.895	7.105	11.608
CLEAN-DS-60-SUMMER	1.816	4.299	4.529	1.829	4.519	5.147
CLEAN-DS-60-WINTER	1.648	4.243	4.678	1.839	4.459	5.443
DIRTY-DS-15-SUMMER	1.912	12.158	19.096	2.009	13.861	22.447
DIRTY-DS-15-WINTER	2.661	11.648	19.100	7.049	29.010	22.319
DIRTY-DS-30-SUMMER	2.500	6.701	9.464	2.621	7.314	10.942
DIRTY-DS-30-WINTER	1.828	6.606	9.504	1.849	7.761	10.723
DIRTY-DS-60-SUMMER	1.783	3.828	3.936	1.859	4.566	5.200
DIRTY-DS-60-WINTER	1.706	4.142	4.628	1.901	5.065	5.188
SMOOTH-DS-15-SUMMER	1.732	11.807	19.284	1.970	14.001	22.177
SMOOTH-DS-15-WINTER	1.722	12.143	19.633	2.066	13.849	22.789
SMOOTH-DS-30-SUMMER	1.690	6.553	10.389	1.922	7.433	11.659
SMOOTH-DS-30-WINTER	1.781	7.347	9.390	1.889	8.221	11.167
SMOOTH-DS-60-SUMMER	1.665	5.113	4.631	1.835	5.823	5.372
SMOOTH-DS-60-WINTER	1.720	5.250	4.600	1.877	5.509	5.426

Table 11
Results of the MLP technique, values in sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination) RMSE (root mean square error) MAE (mean absolute error). RMSE and MAE are measured in degrees Celsius (°C).

Forecasting period	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.588_{0.120}$	$4.044_{0.287}$	$3.710_{0.267}$	$0.806_{0.053}$	$3.366_{0.191}$	$2.932_{0.201}$
CLEAN-DS-15-WINTER	$0.838_{0.038}$	$1.538_{0.081}$	$1.276_{0.093}$	$0.886_{0.018}$	$1.643_{0.088}$	$1.279_{0.079}$
CLEAN-DS-30-SUMMER	$0.626_{0.119}$	$3.768_{0.222}$	$3.434_{0.260}$	$0.821_{0.032}$	$3.242_{0.232}$	$2.809_{0.244}$
CLEAN-DS-30-WINTER	$0.853_{0.043}$	$1.634_{0.141}$	$1.403_{0.135}$	$0.891_{0.027}$	$1.706_{0.082}$	$1.348_{0.062}$
CLEAN-DS-60-SUMMER	$0.835_{0.100}$	$3.242_{0.286}$	$2.927_{0.289}$	$0.878_{0.043}$	$3.327_{0.133}$	$2.962_{0.151}$
CLEAN-DS-60-WINTER	$0.830_{0.024}$	$1.708_{0.064}$	$1.506_{0.066}$	$0.899_{0.008}$	$1.654_{0.027}$	$1.327_{0.028}$
DIRTY-DS-15-SUMMER	$0.565_{0.147}$	$3.906_{0.413}$	$3.528_{0.405}$	$0.774_{0.093}$	$3.301_{0.291}$	$2.832_{0.270}$
DIRTY-DS-15-WINTER	$0.816_{0.037}$	$1.519_{0.117}$	$1.278_{0.127}$	$0.876_{0.012}$	$1.578_{0.104}$	$1.262_{0.094}$
DIRTY-DS-30-SUMMER	$0.613_{0.081}$	$3.858_{0.255}$	$3.529_{0.264}$	$0.821_{0.034}$	$3.147_{0.132}$	$2.701_{0.113}$
DIRTY-DS-30-WINTER	$0.787_{0.052}$	$1.728_{0.133}$	$1.488_{0.117}$	$0.870_{0.029}$	$1.677_{0.102}$	$1.357_{0.081}$
DIRTY-DS-60-SUMMER	$0.819_{0.075}$	$3.136_{0.333}$	$2.833_{0.354}$	$0.877_{0.041}$	$3.154_{0.355}$	$2.832_{0.343}$
DIRTY-DS-60-WINTER	$0.828_{0.018}$	$1.737_{0.073}$	$1.533_{0.069}$	$0.900_{0.008}$	$1.635_{0.031}$	$1.322_{0.038}$
SMOOTH-DS-15-SUMMER	$0.685_{0.149}$	$3.781_{0.405}$	$3.494_{0.374}$	$0.849_{0.056}$	$3.387_{0.196}$	$2.995_{0.217}$
SMOOTH-DS-15-WINTER	$0.790_{0.065}$	$1.599_{0.209}$	$1.297_{0.232}$	$0.867_{0.031}$	$1.741_{0.125}$	$1.338_{0.137}$
SMOOTH-DS-30-SUMMER	$0.705_{0.125}$	$3.775_{0.358}$	$3.451_{0.372}$	$0.855_{0.039}$	$3.395_{0.165}$	$3.000_{0.139}$
SMOOTH-DS-30-WINTER	$0.784_{0.043}$	$1.763_{0.182}$	$1.464_{0.197}$	$0.875_{0.021}$	$1.877_{0.100}$	$1.456_{0.087}$
SMOOTH-DS-60-SUMMER	$0.748_{0.077}$	$3.457_{0.160}$	$3.171_{0.143}$	$0.895_{0.027}$	$3.662_{0.227}$	$3.261_{0.214}$
SMOOTH-DS-60-WINTER	$0.722_{0.040}$	$2.012_{0.129}$	$1.733_{0.158}$	$0.854_{0.020}$	$2.127_{0.065}$	$1.718_{0.071}$

Contrary to the MLP technique, although with really little difference, the best result is obtained with a dataset without preprocessing. This makes sense as deep learning techniques are better able to remove possible noise in the data.

Table 13 shows the results of the LSTM technique. This technique obtains a similar behavior to the previous deep learning techniques. In the overall tuning, there is no difference in the prediction at 12 or 24 h. Moreover, although the results are similar, this technique obtains slightly higher RMSE and MAE values than the other deep learning techniques discussed. The best results are obtained with the DIRTY-DS-60-WINTER dataset, where its MAE and RMSE are 1.294 °C and 1.554 °C in 12 h prediction respectively.

Finally, Table 14 shows the quality figures of the CNNLSTM technique, which reports very similar results to the CNN technique. It follows the trend of obtaining very similar results for 12 h and 24 h forecasts. Moreover, the best result is obtained with the DIRTY-DS-60-WINTER dataset. In this case, although also with little difference, the best result is obtained with the data without preprocessing. Specifically, the MAE and RMSE results are 1.200 °C and 1.357 °C respectively.

4.3. Model retraining evaluation

ML models require periodic retraining to improve the quality of predictions as new data is generated. However, these retrainings are quite computationally expensive, and that is why SEPARATE allows asynchronous communication with an HPC server that

Table 12
Results of the CNN technique, values in sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination) RMSE (root mean square error) MAE (mean absolute error). RMSE and MAE are measured in degrees Celsius (°C).

Forecasting period	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.739_{0.189}$	$3.605_{0.286}$	$3.279_{0.208}$	$0.867_{0.044}$	$3.086_{0.137}$	$2.682_{0.158}$
CLEAN-DS-15-WINTER	$0.789_{0.224}$	$1.782_{0.831}$	$1.545_{0.855}$	$0.808_{0.283}$	$1.787_{0.753}$	$1.463_{0.753}$
CLEAN-DS-30-SUMMER	$0.815_{0.174}$	$3.499_{0.255}$	$3.234_{0.194}$	$0.897_{0.049}$	$3.267_{0.052}$	$2.906_{0.100}$
CLEAN-DS-30-WINTER	$0.778_{0.022}$	$1.549_{0.126}$	$1.259_{0.095}$	$0.876_{0.010}$	$1.613_{0.041}$	$1.257_{0.038}$
CLEAN-DS-60-SUMMER	$0.841_{0.022}$	$3.475_{0.070}$	$3.200_{0.096}$	$0.859_{0.008}$	$3.360_{0.056}$	$3.020_{0.059}$
CLEAN-DS-60-WINTER	$0.829_{0.013}$	$1.723_{0.022}$	$1.503_{0.025}$	$0.899_{0.004}$	$1.627_{0.012}$	$1.311_{0.014}$
DIRTY-DS-15-SUMMER	$0.799_{0.102}$	$3.347_{0.106}$	$3.014_{0.153}$	$0.862_{0.025}$	$3.034_{0.174}$	$2.642_{0.162}$
DIRTY-DS-15-WINTER	$0.795_{0.208}$	$1.834_{0.782}$	$1.597_{0.805}$	$0.808_{0.282}$	$1.859_{0.721}$	$1.510_{0.727}$
DIRTY-DS-30-SUMMER	$0.870_{0.074}$	$3.431_{0.187}$	$3.197_{0.165}$	$0.898_{0.040}$	$3.160_{0.073}$	$2.822_{0.049}$
DIRTY-DS-30-WINTER	$0.814_{0.025}$	$1.492_{0.201}$	$1.252_{0.210}$	$0.886_{0.008}$	$1.539_{0.102}$	$1.222_{0.098}$
DIRTY-DS-60-SUMMER	$0.789_{0.029}$	$3.553_{0.075}$	$3.299_{0.088}$	$0.853_{0.008}$	$3.249_{0.046}$	$2.890_{0.056}$
DIRTY-DS-60-WINTER	$0.831_{0.013}$	$1.666_{0.027}$	$1.442_{0.034}$	$0.898_{0.005}$	$1.600_{0.020}$	$1.281_{0.016}$
SMOOTH-DS-15-SUMMER	$0.745_{0.160}$	$3.535_{0.246}$	$3.161_{0.269}$	$0.871_{0.036}$	$3.026_{0.166}$	$2.598_{0.214}$
SMOOTH-DS-15-WINTER	$0.681_{0.310}$	$2.058_{1.008}$	$1.818_{1.044}$	$0.708_{0.373}$	$2.064_{0.955}$	$1.730_{0.958}$
SMOOTH-DS-30-SUMMER	$0.754_{0.156}$	$3.530_{0.182}$	$3.239_{0.161}$	$0.893_{0.053}$	$3.338_{0.095}$	$2.980_{0.122}$
SMOOTH-DS-30-WINTER	$0.756_{0.012}$	$1.745_{0.095}$	$1.443_{0.104}$	$0.863_{0.004}$	$1.793_{0.027}$	$1.401_{0.027}$
SMOOTH-DS-60-SUMMER	$0.846_{0.018}$	$3.482_{0.120}$	$3.249_{0.135}$	$0.933_{0.004}$	$3.617_{0.080}$	$3.249_{0.062}$
SMOOTH-DS-60-WINTER	$0.696_{0.012}$	$2.039_{0.046}$	$1.752_{0.050}$	$0.840_{0.006}$	$2.157_{0.019}$	$1.745_{0.017}$

Table 13
Results of the LSTM technique, values in sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination) RMSE (root mean square error) MAE (mean absolute error). RMSE and MAE are measured in degrees Celsius (°C).

Prediction hours	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.605_{0.260}$	$3.602_{0.550}$	$2.501_{0.242}$	$0.521_{0.324}$	$5.819_{1.225}$	$4.773_{0.925}$
CLEAN-DS-15-WINTER	$0.257_{0.057}$	$2.785_{0.043}$	$2.073_{0.056}$	$0.323_{0.058}$	$3.985_{0.167}$	$2.967_{0.064}$
CLEAN-DS-30-SUMMER	$0.836_{0.052}$	$2.658_{0.209}$	$2.209_{0.231}$	$0.912_{0.035}$	$2.813_{0.268}$	$2.451_{0.229}$
CLEAN-DS-30-WINTER	$0.779_{0.041}$	$1.724_{0.196}$	$1.400_{0.185}$	$0.875_{0.029}$	$1.719_{0.124}$	$1.367_{0.107}$
CLEAN-DS-60-SUMMER	$0.827_{0.026}$	$2.756_{0.203}$	$2.401_{0.270}$	$0.930_{0.012}$	$3.000_{0.192}$	$2.646_{0.173}$
CLEAN-DS-60-WINTER	$0.796_{0.022}$	$1.655_{0.154}$	$1.389_{0.153}$	$0.893_{0.010}$	$1.609_{0.088}$	$1.258_{0.079}$
DIRTY-DS-15-SUMMER	$0.481_{0.234}$	$4.324_{1.610}$	$3.151_{1.026}$	$0.530_{0.234}$	$6.649_{2.812}$	$5.518_{2.373}$
DIRTY-DS-15-WINTER	$0.288_{0.109}$	$3.088_{0.256}$	$2.611_{0.353}$	$0.318_{0.136}$	$3.516_{0.408}$	$2.905_{0.191}$
DIRTY-DS-30-SUMMER	$0.805_{0.051}$	$2.688_{0.199}$	$2.275_{0.214}$	$0.909_{0.024}$	$2.803_{0.216}$	$2.460_{0.189}$
DIRTY-DS-30-WINTER	$0.751_{0.077}$	$1.885_{0.375}$	$1.540_{0.340}$	$0.857_{0.054}$	$1.735_{0.248}$	$1.366_{0.217}$
DIRTY-DS-60-SUMMER	$0.802_{0.030}$	$3.000_{0.199}$	$2.691_{0.175}$	$0.927_{0.009}$	$2.808_{0.107}$	$2.466_{0.136}$
DIRTY-DS-60-WINTER	$0.785_{0.057}$	$1.554_{0.209}$	$1.294_{0.171}$	$0.881_{0.041}$	$1.656_{0.102}$	$1.317_{0.096}$
SMOOTH-DS-15-SUMMER	$0.550_{0.269}$	$4.771_{2.427}$	$3.614_{1.750}$	$0.562_{0.171}$	$6.450_{3.994}$	$5.426_{3.485}$
SMOOTH-DS-15-WINTER	$0.401_{0.116}$	$2.936_{0.544}$	$2.555_{0.547}$	$0.385_{0.026}$	$3.236_{0.156}$	$2.788_{0.203}$
SMOOTH-DS-30-SUMMER	$0.711_{0.062}$	$3.215_{0.295}$	$2.689_{0.219}$	$0.803_{0.107}$	$3.968_{0.947}$	$3.348_{0.744}$
SMOOTH-DS-30-WINTER	$0.648_{0.166}$	$2.124_{0.277}$	$1.744_{0.198}$	$0.783_{0.175}$	$2.115_{0.592}$	$1.669_{0.417}$
SMOOTH-DS-60-SUMMER	$0.833_{0.019}$	$2.982_{0.084}$	$2.663_{0.107}$	$0.934_{0.008}$	$3.419_{0.246}$	$3.029_{0.224}$
SMOOTH-DS-60-WINTER	$0.665_{0.027}$	$2.102_{0.115}$	$1.779_{0.111}$	$0.823_{0.016}$	$2.153_{0.032}$	$1.731_{0.039}$

retrains the model and, once retrained, can be updated at the edge. The question here was how often it is necessary to perform these trainings. To analyze this point, the following tests have been carried out. First, the model has been trained using all the data from each dataset under study, except for the last 30 days. Subsequently, the model was tested with day 1, then day 2, then day 3, and so on until day 30. Independent tests were performed on each day, in order to visualize and analyze the impact of error and model fit as the days passed. The results of these tests indicate the maximum time that an ML/DL time series model can predict without retraining or loss of prediction quality. Although all techniques have similar performance, the one with the lowest error is the CNNLTSM, so this test has been performed with this technique.

Figs. 4(a) and 4(b) show R^2 and the mean of RMSE and MAE metrics for all targeted datasets. Particularly, Fig. 4(a) reports significant sharp differences testing 10, 16, 20, and 21 days respectively using R^2 metric. Fig. 4(b) focuses on RMSE and MAE metrics and it can be seen that from days 9 and 10 onwards, there is a significant upward trend in the error. The same occurs on days 16, 20, and 21, the latter two being the days with the greatest error with respect to the rest of the days.

After showing the test results evaluating 30 days separately and analyzing that days 9 and 10, 16, 20, and 21 seem to be days with significant key differences, we now apply non-parametric statistical tests to statistically contract from which day the CNNLSTM model should be retrained due to the significant increase in error and the drop in fit. We perform a non-parametric statistical test, in particular, the Kruskal–Wallis test [29], since the data do not follow a normal distribution. To make this statement, we proceeded to perform a Kolmogorov–Smirnov test [30] for normality of the data, obtaining a p-value of 0.000, which indicates that the RMSE, MAE, and R^2 data do not follow a normal distribution. Kruskal–Wallis tests have been also performed using all datasets, without

Table 14
Results of the CNNLSTM technique, values in sub-index indicate the standard deviation obtained after the repetition of each experiment. R^2 (coefficient of determination) RMSE (root mean square error) MAE (mean absolute error). RMSE and MAE are measured in degrees Celsius (°C).

Prediction hours	12 h			24 h		
Datasets	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}	R^2_{sd}	$RMSE_{sd}$	MAE_{sd}
CLEAN-DS-15-SUMMER	$0.786_{0.088}$	$3.344_{0.179}$	$3.014_{0.203}$	$0.876_{0.023}$	$2.947_{0.321}$	$2.553_{0.368}$
CLEAN-DS-15-WINTER	$0.875_{0.023}$	$1.527_{0.185}$	$1.303_{0.193}$	$0.923_{0.008}$	$1.479_{0.150}$	$1.193_{0.110}$
CLEAN-DS-30-SUMMER	$0.794_{0.106}$	$3.483_{0.251}$	$3.210_{0.225}$	$0.860_{0.051}$	$3.223_{0.067}$	$2.843_{0.079}$
CLEAN-DS-30-WINTER	$0.874_{0.017}$	$1.599_{0.087}$	$1.401_{0.091}$	$0.921_{0.007}$	$1.595_{0.035}$	$1.280_{0.032}$
CLEAN-DS-60-SUMMER	$0.935_{0.005}$	$3.102_{0.039}$	$2.867_{0.058}$	$0.939_{0.010}$	$3.291_{0.063}$	$2.956_{0.051}$
CLEAN-DS-60-WINTER	$0.876_{0.003}$	$1.441_{0.023}$	$1.281_{0.022}$	$0.930_{0.001}$	$1.504_{0.040}$	$1.220_{0.018}$
DIRTY-DS-15-SUMMER	$0.827_{0.045}$	$3.381_{0.222}$	$3.092_{0.244}$	$0.863_{0.024}$	$2.975_{0.194}$	$2.595_{0.196}$
DIRTY-DS-15-WINTER	$0.881_{0.025}$	$1.527_{0.111}$	$1.295_{0.124}$	$0.927_{0.009}$	$1.594_{0.120}$	$1.258_{0.102}$
DIRTY-DS-30-SUMMER	$0.868_{0.097}$	$3.359_{0.172}$	$2.887_{0.298}$	$0.857_{0.029}$	$3.575_{0.364}$	$3.142_{0.296}$
DIRTY-DS-30-WINTER	$0.877_{0.009}$	$1.575_{0.052}$	$1.370_{0.051}$	$0.926_{0.003}$	$1.593_{0.057}$	$1.267_{0.040}$
DIRTY-DS-60-SUMMER	$0.937_{0.004}$	$3.093_{0.038}$	$2.866_{0.061}$	$0.942_{0.004}$	$3.278_{0.058}$	$2.952_{0.053}$
DIRTY-DS-60-WINTER	$0.882_{0.006}$	$1.357_{0.043}$	$1.200_{0.048}$	$0.935_{0.003}$	$1.404_{0.018}$	$1.142_{0.022}$
SMOOTH-DS-15-SUMMER	$0.714_{0.079}$	$3.358_{0.315}$	$2.994_{0.366}$	$0.844_{0.104}$	$2.904_{0.346}$	$2.457_{0.335}$
SMOOTH-DS-15-WINTER	$0.848_{0.038}$	$1.630_{0.199}$	$1.407_{0.195}$	$0.904_{0.016}$	$1.596_{0.134}$	$1.283_{0.107}$
SMOOTH-DS-30-SUMMER	$0.776_{0.143}$	$3.528_{0.271}$	$3.253_{0.242}$	$0.893_{0.051}$	$3.281_{0.062}$	$2.903_{0.072}$
SMOOTH-DS-30-WINTER	$0.787_{0.040}$	$1.787_{0.093}$	$1.506_{0.119}$	$0.888_{0.016}$	$1.832_{0.062}$	$1.435_{0.032}$
SMOOTH-DS-60-SUMMER	$0.906_{0.012}$	$3.369_{0.050}$	$3.133_{0.119}$	$0.947_{0.008}$	$3.763_{0.094}$	$3.375_{0.067}$
SMOOTH-DS-60-WINTER	$0.715_{0.015}$	$2.010_{0.051}$	$1.732_{0.055}$	$0.854_{0.008}$	$2.110_{0.060}$	$1.724_{0.033}$

Table 15
P-values of the statistical analysis for all datasets, considering the MAE metric of each of the tested days. Only *p*-values < 0.06 are shown.

Days	1	2	3	4	5	6	7	8	9	10
9	0.002	0.054								
10	0.002	0.051								
12	0.011									
16	0.000	0.000								
17	0.000	0.000								
18	0.000	0.007								
19	0.000	0.011								
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.080	0.007
21	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.011

Table 16
P-values of the statistical analysis for all datasets, considering the RMSE metric of each of the tested days. Only *p*-values < 0.06 are shown.

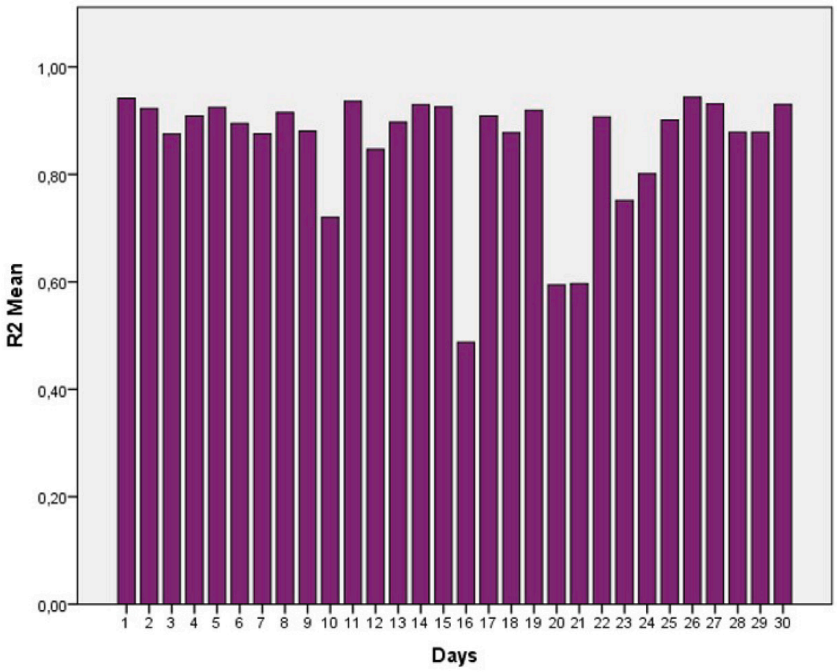
days	1	2	3	4	5	6	7	8	10
9	0.000	0.002							
10	0.002	0.043							
12	0.012								
16	0.000	0.000							
17	0.000	0.000							
18	0.000	0.004							
19	0.001	0.002							
20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007
21	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.011

distinguishing by data pre-processing used, the season of the year, as well as granularity. For a greater amount of information, tests have been performed individually for each of the metrics evaluated in the accuracy, which have been MSE, RMSE, and R^2. The statistical results, with a 95% confidence level, after performing the Kruskal–Wallis test, show the adjusted *p*-values indicated in Tables 15–17.

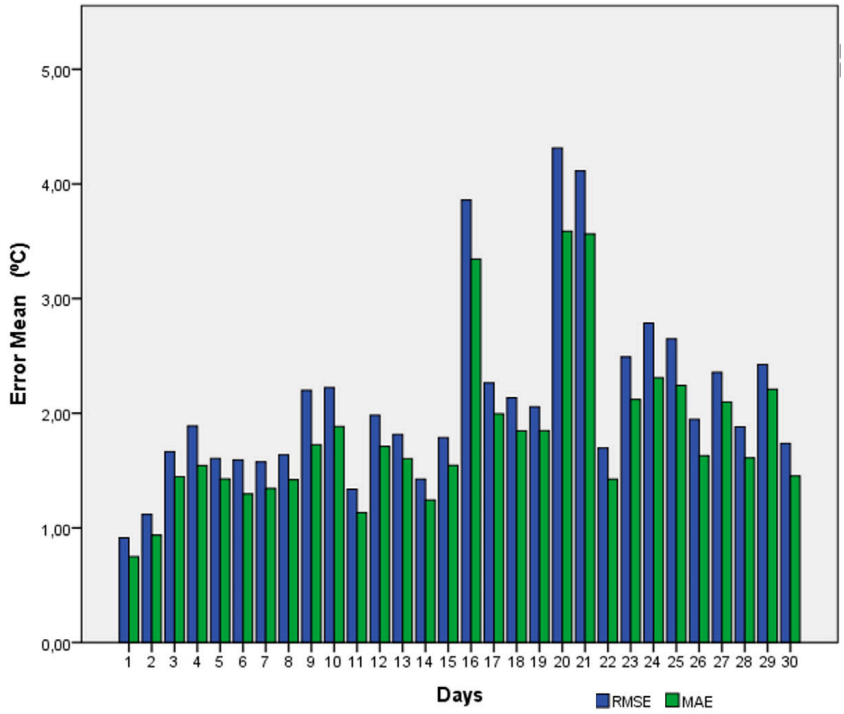
Table 15 shows for the first 10 days, the *p*-values with values higher than 0.05; i.e., confidence level of 95%. Thus, it is important to note that the first two days have significant differences with a 95% confidence level, with days 9, 10, 12, 16, 17, 18, 19, 20, and 21. Days 3 to 10, however, have significant differences from the models for days 20 and 21. These results lead us to conclude that taking into account the MAE metric, from day 9 onwards, the model starts to lose accuracy, however, when the accuracy drops drastically is from day 19 onwards.

Table 16 shows the *p*-values, with a confidence level of 95%, for the first 10 days for the RMSE metric. As can be seen, the results are very similar to the MAE metric, with the only exception that day 9 has no significant differences with days 20 and 21. Despite this difference, the conclusion is the same as that obtained for the MAE metric, from day 9 onwards, accuracy decreases but drops drastically from day 19 onwards. Therefore, the two error metrics indicate the same conclusion.

The 17 shows the *p*-values, at 95% confidence level, for the values of the R2 metric. It shows significant differences in change with respect to errors. It should be considered that the R2 metric measures the overall model fit, so it is possible that it does not



(a) Mean of all datasets in table 1 for the metric R^2. The Y-axis shows the correlation between the real and the forecasted values expressed in a range between 0 and 1.



(b) Mean of all datasets in table 1 for the metric RMSE y MAE. The Y-axis shows the mean error expressed in degrees Celsius.

Fig. 4. Mean of all datasets in Table 1 for all the metrics.

represent the errors as much as the overall fit. This table also shows how the days are reduced, and there are significant differences in the R2 metric of days 1 and 2 with the models of days 10, 12, 16, 20, 21. While for days 4, 5, 7, and 8 there are only significant differences with the models for days 20 and 21. These results lead us to the same conclusion as indicated by the previous metrics.

Table 17
P-values of the statistical analysis for all datasets, considering the R^2 metric of each of the tested days. Only *p*-values < 0.06 are shown.

Days	1	2	4	5	7	8
10	0.000	0.000	0.000	0.000	0.000	0.000
12	0.001					
16	0.005					
20	0.000	0.000	0.025	0.000		0.002
21	0.000	0.000	0.024	0.000		0.002

From day 9 onwards, the accuracy and fit of the models drop, on the tenth-day accuracy is lost. However, when the accuracy drops drastically from day 20 onwards.

Thus, we can conclude that days 1 and 2 are the most accurate, but there is really no significant loss of accuracy until day 9. Days 20 and 21 have significant differences with the first 10 days. This already indicates that from then on, the model decreases in accuracy, so from day 19, if day 9 has not been carried out, the model should be retrained.

The problem of retraining can be solved by using incremental learning. Although there is no common framework for dealing with this problem in deep learning techniques, there are some publications that address it. Incremental learning can be classified into three scenarios [31]: Task-incremental learning, Domain-incremental learning and Class-incremental learning. Task-incremental learning scenario is about learning sequentially to solve a series of tasks in the same context. Domain-incremental learning scenario focuses on solving the same problem but in different contexts. In contrast, Class-incremental learning focuses on learning to discriminate between incrementally observed classes. In the case we are concerned with in this study, we will be able to perform incremental learning to change context, for example, to change the greenhouse, or to perform incremental tasks, which would be to continue predicting the temperature in the same greenhouse. Some publications perform incremental learning using Deep Learning techniques. Thus in [32] the authors present an incremental learning method using an LSTM network for attitude estimation of an object in space using a gyroscope, accelerometer, and magnetometer sensors. It is initially trained with sensor data that is dynamically updated at runtime by the LSTM network. The incremental learning is done in a rough way, starting from the initial weights of the trained network and only training with the new information available. Another incremental learning technique based on an LSTM is presented in [33], in this case, it is not focused on a specific problem, but the technique is evaluated with a synthetic data set and good results are obtained. Another study that also uses Deep learning techniques adapted to computational learning is presented in [34]. The authors present an incremental learning paradigm called Deep Model Consolidation to learn labels when the original training data is not available. The idea of the work is to first train a separate model for new labels only, and then combine the two individual models trained on data from two different sets of labels (old labels and new labels) through a novel double distillation training objective. Therefore, in view of the results obtained in the literature, the problem of retraining is solved by applying incremental learning, although future work may study in depth specific techniques for the problem addressed in this study.

5. Discussion

This section first analyzes the proposed Pub/Sub solution (see Section 3.2) using all the metrics obtained in Sections 4.1.1 and 4.1.2. Moreover, the precision of the ML/DL models used (see Section 3.4) is analyzed using metrics obtained in Section 4.2. Finally, the necessary retraining time is discussed following the metrics shown in Section 4.3.

Regarding the Pub/Sub solution, two aspects should be considered: (1) the execution time spent on training and inference with the different ML/DL models, executed on CPU or GPU in the HPC computing platform; and (2) the execution time spent on loading and forecasting with the different ML/DL models, executed on CPU and GPU of the edge computing platform.

HPC platforms are needed for training heavy models, like those ones used in this investigation. Analyzing the obtained results can be concluded that using GPU is much faster than using only a CPU obtaining, on average, a 1.25X performance. However, in terms of inference, both CPU and GPU are very evenly matched, with the CPU being slightly faster than the GPU (on average, 0.94X of performance). This makes sense in our case because, although the models are heavy for training as they use large datasets to do this task, the inference is reduced to the univariate prediction of the greenhouse internal temperature. Furthermore, a maximum prediction horizon of 24 h is considered, as a larger time horizon is not operational in the greenhouse. However, this implies that at the smallest temporal granularity; i.e. 15 min, no more than 96 values are predicted, which implies a light forecast for this type of deep learning model.

Edge computing is needed to generate a forecast as soon as data is generated or received. Analyzing the obtained results can be concluded that using CPU is much faster than using GPU obtaining, on average, a 2.44X performance as far as the model load is concerned. However, in terms of prediction, both CPU and GPU are evenly matched, with the CPU being slightly faster than the GPU (on average, 0.91X of performance). As with HPC platforms, this makes sense in our case because, although the stored models are heavy, once loaded into memory it is much faster to perform the inference on CPU rather than having to take it to GPU since the inference is still reduced to the univariate prediction of the internal greenhouse temperature as in the previous case. Although the model file is heavy, the model loading and univariate prediction tasks are very light tasks, so performing them on GPU means the appearance of bottlenecks, especially in memory access tasks, not taking full advantage of the GPU's potential.

Comparing the model training/loading and inference times on HPC and edge platforms, it can be concluded that loading the model in an edge platform is much faster than training it in an HPC platform (on average, 1,057.71X performance). Moreover, due to the low computational capabilities of edge platforms, the forecasting time in edge platforms is much slower than in HPC platforms (on average, 0.24X performance).

Regarding the accuracy of ML/DL models, it can be concluded that it depends mainly on the characteristics of the data. For example, when there is a lot of data with a low temporality (such as 15 min datasets), models such as CNN and CNNLSTM perform very well. However, when the data has a higher temporal granularity (such as 60 min datasets), models such as LSTM show the best performance. It should also be noted that there are simple (in terms of architecture) models such as MLP that have reported good results. In addition, it is important to note that smooth pre-processing is not effective for any of the techniques, as it always provides lower performance. Moreover, for DL techniques, the best results are usually obtained with the raw datasets, as they are able to remove possible noise from the data.

Although there are no major differences with the other techniques, the most stable technique is CNNLSTM because its accuracy performance is somewhat better than the other techniques, at both 12 and 24 h, regardless of the temporal granularity of the data. However, in terms of computing time, it is one of the slowest techniques for inference, although the times are still manageable, so given the importance of greenhouse temperature and the fact that the time difference is only 1 or 2 s, it can be selected as the best technique to implement.

Finally, we discuss the analysis of the days needed to perform model retraining. Considering the results shown, in the first 9 days, the model remains stable in all metrics. From day 9 onwards, the model loses accuracy. However, the results have shown that when there is really a significant loss of accuracy is from day 20 onwards, as all previous days have significant differences with the metrics obtained from day 20. This shows the need for retraining on the ninth day of the model run. If there are any problems in the system, there would be a margin of 11 days to retrain the model (i.e., until day 20), where the model update should be mandatory. Thus, we can conclude that from day 19 the model should collect all the data and retrain the new model without delay, although the best day not to lose accuracy would be day 9. It is important to consider that from day 9 the model must be trained, if we are looking for optimal accuracy, for any kind of temporal granularity, the season of the year, as well as data pre-processing.

6. Conclusions and future work

New technologies are helping to improve quality, efficiency, and profitability in many areas. Farming in recent years is gaining a great deal of performance when new technologies bring their advantages. Proof of this is the automation that is taking place in greenhouses. IoT and new communications protocols help to connect sensors, with information-gathering and decision-making systems quickly and effectively, achieving fast and efficient monitoring and execution of actions. Thus, in this research, an infrastructure applied to an operational greenhouse has been proposed, consisting of a Pub/Sub-based infrastructure that offers an interoperable and decentralized dynamic architecture for ML/DL training and inference. The infrastructure is evaluated to forecast the internal temperature of an operational greenhouse. This allows the farmer to act in advance to have the best climatic conditions for his crops, with a reliable and fast-running model on the edge. After this research, it can be concluded that Pub/Sub systems are nowadays mandatory for IoT applications that require taking actions in almost real-time. When a real-time data publisher is considered to monitoring an ecosystem, and decisions need to be taken based on analytics and forecast, it is mandatory to have a tightly coupled, seamless IoT infrastructure that covers the whole data cycle from the capture to the processing pipeline to analyze them and make reliable forecasting in the shortest possible time in order to make decisions in NRT. Finally, it should be noted that the best prediction model found for the SEPARATE infrastructure was the CNNLTSM model. Moreover, it has been analyzed and from the 9th day of prediction at any granularity and/or season of the year, the model should be retrained in order not to lose accuracy.

The future work of this research consists of optimizing the developed architecture, following different strategies that could improve the overall performance such as analyzing other possible Pub/Sub architectures that improve the response times of the proposal. Moreover, in order to optimize the models by which this architecture is developed, the following strategies could be studied: incorporating other artificial intelligence models different from the existing ones; increasing the number of variables to be predicted using the developed architecture; and changing the training strategy (batch training) for one that allows optimizing the training time, such as online/incremental training.

CRediT authorship contribution statement

Juan Morales-García: Methodology, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Andrés Bueno-Crespo:** Methodology, Validation, Writing – original draft, Writing – review & editing, Funding acquisition. **Raquel Martínez-España:** Methodology, Validation, Writing – original draft, Writing – review & editing, Visualization. **Francisco J. García:** Conceptualization, Formal analysis. **Sergio Ros:** Conceptualization, Formal analysis. **Julio Fernández-Pedauyé:** Software. **José M. Cecilia:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jose M. Cecilia, Andrés Bueno-Crespo, Juan Morales-García, Francisco J. García and Sergio Navarro report financial support was provided by Spanish Ministry of Science.

Data availability

Data will be made available on request.

Funding

This work is derived from R&D projects RTC2019-007159-5, as well as the Ramon y Cajal Grant RYC2018-025580-I, funded by MCIN/AEI/10.13039/501100011033, "FSE invest in your future" and "ERDF A way of making Europe". All authors approved the version of the manuscript to be published.

References

[1] J. Lowenberg-DeBoer, The economics of precision agriculture, in: Precision Agriculture for Sustainability, Burleigh Dodds Science Publishing, 2019, pp. 481–502.
[2] M.A. Zamora-Izquierdo, J. Santa, J.A. Martínez, V. Martínez, A.F. Skarmeta, Smart farming IoT platform based on edge and cloud computing, Biosyst. Eng. 177 (2019) 4–17.
[3] M.C. Garrido, J.M. Cadenas, A. Bueno-Crespo, R. Martínez-España, J.G. Giménez, J.M. Cecilia, Evaporation forecasting through interpretable data analysis techniques, Electronics 11 (4) (2022) 536.
[4] D.A. Howard, Z. Ma, C. Veje, A. Clausen, J.M. Aaslyng, B.N. Jørgensen, Greenhouse industry 4.0–digital twin technology for commercial greenhouses, Energy Inform. 4 (2) (2021) 1–13.
[5] J. Yang, A. Sharma, R. Kumar, IoT-based framework for smart agriculture, Int. J. Agric. Environ. Inf. Syst. (IJAEIS) 12 (2) (2021) 1–14.
[6] E. Yaacoub, M.-S. Alouini, A key 6G challenge and opportunity—Connecting the base of the pyramid: A survey on rural connectivity, Proc. IEEE 108 (4) (2020) 533–582.
[7] K.G. Liakos, P. Busato, D. Moshou, S. Pearson, D. Bochtis, Machine learning in agriculture: A review, Sensors 18 (8) (2018) 2674.
[8] A. Kamilaris, F.X. Prenafeta-Boldú, Deep learning in agriculture: A survey, Comput. Electron. Agric. 147 (2018) 70–90.
[9] D. Garg, M. Alam, Deep learning and IoT for agricultural applications, in: Internet of Things (IoT), Springer, 2020, pp. 273–284.
[10] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39.
[11] P. Warden, D. Situnayake, TinyML, O'Reilly Media, Incorporated, 2019.
[12] Y. Wu, E. Dobriban, S. Davidson, Deltagrad: Rapid retraining of machine learning models, in: International Conference on Machine Learning, PMLR, 2020, pp. 10355–10366.
[13] B. Mishra, A. Kertesz, The use of MQTT in M2M and IoT systems: A survey, IEEE Access 8 (2020) 201071–201086.
[14] B. Wukkadada, K. Wankhede, R. Nambiar, A. Nair, Comparison with HTTP and MQTT in internet of things (IoT), in: 2018 International Conference on Inventive Research in Computing Applications, ICIRCA, IEEE, 2018, pp. 249–253.
[15] S.-M. Kim, H.-S. Choi, W.-S. Rhee, IoT home gateway for auto-configuration and management of MQTT devices, in: 2015 IEEE Conference on Wireless Sensors, ICWiSe, IEEE, 2015, pp. 12–17.
[16] N. Tantitharanukul, K. Osathanunkul, K. Hantrakul, P. Pramokchon, P. Khoenkaw, MQTT-topics management system for sharing of open data, in: 2017 International Conference on Digital Arts, Media and Technology, ICDAMT, IEEE, 2017, pp. 62–65.
[17] Y. Syafarinda, F. Akhadin, Z. Fitri, B. Widiawan, E. Rosdiana, et al., The precision agriculture based on wireless sensor network with MQTT protocol, in: IOP Conference Series: Earth and Environmental Science, Vol. 207, No. 1, IOP Publishing, 2018, 012059.
[18] A.A. Ahmed, S. Al Omari, R. Awal, A. Fares, M. Chouikha, A distributed system for supporting smart irrigation using internet of things technology, Eng. Rep. 3 (7) (2021).
[19] F.M. Taha, A.A. Osman, S.D. Awadalkareem, M.S. Omer, R.S. Saadaldeen, A design of a remote greenhouse monitoring and controlling system based on internet of things, in: 2018 International Conference on Computer, Control, Electrical, and Electronics Engineering, ICCCEEE, IEEE, 2018, pp. 1–6.
[20] T.A. Singh, J. Chandra, IOT based green house monitoring system, J. Comput. Sci. 14 (5) (2018) 639–644.
[21] R. Tashakkori, A.S. Hamza, M.B. Crawford, Beemon: An IoT-based beehive monitoring system, Comput. Electron. Agric. 190 (2021) 106427.
[22] U. Hunkeler, H.L. Truong, A. Stanford-Clark, MQTT-S—A publish/subscribe protocol for wireless sensor networks, in: 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops, COMSWARE'08, IEEE, 2008, pp. 791–798.
[23] M. Bender, E. Kirdan, M.-O. Pahl, G. Carle, Open-source mqtt evaluation, in: 2021 IEEE 18th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2021, pp. 1–4.
[24] J. Brownlee, Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, Machine Learning Mastery, 2018.
[25] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[26] S. Haykin, R. Lippmann, Neural Networks: A Comprehensive Foundation, second ed., Prentice Hall PTR, 1998.
[27] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[28] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[29] T.V. Hecke, Power study of anova versus Kruskal-Wallis test, J. Stat. Manag. Syst. 15 (2–3) (2012) 241–247.
[30] G. Marsaglia, W.W. Tsang, J. Wang, Evaluating Kolmogorov's distribution, J. Stat. Softw. 8 (2003) 1–4.
[31] G.M. van de Ven, T. Tuytelaars, A.S. Tolias, Three types of incremental learning, Nat. Mach. Intell. (2022) 1–13.
[32] P. Narkhede, R. Walambe, S. Poddar, K. Kotecha, Incremental learning of LSTM framework for sensor fusion in attitude estimation, PeerJ Comput. Sci. 7 (2021) e662.
[33] Á.C. Lemos Neto, R.A. Coelho, C.L.d. Castro, An incremental learning approach using long short-term memory neural networks, J. Control Autom. Electr. Syst. 33 (5) (2022) 1457–1465.
[34] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, C.-C.J. Kuo, Class-incremental learning via deep model consolidation, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 1131–1140.

2.4. Using remote GPU virtualization techniques to enhance edge computing devices

Los detalles de la cuarta publicación del compendio se muestran en la tabla 4.

Detalles del artículo	
Título	Using remote GPU virtualization techniques to enhance edge computing devices
Revista	Future Generation Computer Systems
Autores	José M. Cecilia, Juan Morales-García, Baldomero Imbernón, Javier Prades, Juan-Carlos Cano, Federico Silla
Año de publicación	2023
DOI	j.future.2022.12.038
Estado	Publicado

Tabla 4: Detalles del cuarto artículo del compendio de publicaciones.

Future Generation Computer Systems 142 (2023) 14–24

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Using remote GPU virtualization techniques to enhance edge computing devices

José M. Cecilia [a], Juan Morales-García [b,*], Baldomero Imbernón [b], Javier Prades [a], Juan-Carlos Cano [a], Federico Silla [a]

[a] Computer and Systems Informatics Department, Universitat Politècnica de València (UPV), Valencia, Spain
[b] Computer Science Department, Catholic University of Murcia (UCAM), Murcia, Spain

ARTICLE INFO

Article history:
Received 2 November 2022
Received in revised form 21 December 2022
Accepted 25 December 2022
Available online 27 December 2022

Keywords:
Machine learning
Clustering algorithms
Edge computing
Remote virtualization
Virtualized GPUs
IoT

ABSTRACT

The Internet of Things (IoT) is driving the next economic revolution where the main actors are both data and immediacy. The IoT ecosystem is increasingly generating large amounts of data that are created but never analyzed. Efficient big data analysis in IoT infrastructures is becoming mandatory to transform this data deluge into meaningful information. Edge computing is proving to be a compelling alternative for enabling computing capabilities at the edge of the network. These computing capabilities could help in transforming the generated data into useful information. However, the edge computing platforms available on the market are low-power devices with limited computing horsepower. In this paper, we present a novel approach to providing computing resources to edge devices without penalizing their power consumption by using remotely virtualized GPUs. We evaluate this hardware environment by executing a computational-intensive clustering algorithm called Fuzzy Minimals (FM). Our results show that using a remotely virtualized GPU on the edge device provides a 3.2x speed-up factor compared to the local counterpart version. Moreover, we report up to 30% reduction in power consumption and up to 80% of energy savings at the edge device, delegating the GPU workload to the backend, transparently to the programmer.

1. Introduction

A key driver of the current digital revolution is the Internet of Things (IoT), where devices and humans are connected to the Internet, interacting with each other in real time. Two key factors that underpin this revolution are (1) data, which carries hidden patterns, correlations, and other valuable insights, and (2) real-time data analytics, required because knowledge is often time-sensitive and useful only within a specific time-frame [1]. The IoT generates zettabytes (10^{21} bytes) of "dark data" which is data never actually analyzed. In addition to the concern about the amount of data, the fast acceleration of data generation further complicates this scenario. Actually, 90% of the data stored in the world was just generated in the last two years [2]. Therefore, enabling efficient big data analytics within the IoT infrastructure is crucial to transform this data deluge into meaningful information.

Machine learning (ML) has become essential for the predictive analysis of huge amounts of data. Additionally, high-performance computing (HPC) plays an equally important role, particularly when the real-time response is crucial. The intersection between ML and HPC is mandatory to deal with large and complex datasets with real-time requirements. In this regard, using a cloud service approach, ML algorithms have been traditionally executed in supercomputers, where performance prevails over energy efficiency [3]. However, when performance is not the only concern, other approaches are feasible. For instance, edge/fog computing [4] is a recent trend towards decentralization, where initial computations on data are carried out close to (or actually at) the capture location. Processing in close proximity to mobile devices or sensors may provide energy savings, highly responsive web services for mobile computing, scalability, and privacy-policy enforcement for the IoT, as well as the ability to mask transient cloud outages. However, IoT devices have a limited power budget at this level of the network, as they typically rely on batteries or energy harvesters, leading to ultra-low power approaches. This limited power scenario translates into a major limitation for many components of the architecture, especially energy-intensive components such as wireless transmitters or even processing capabilities [5].

Edge devices used in IoT usually rely on ultra-low power solutions such as ARM-based CPUs or onboard microcontroller units. Microprocessor companies are releasing systems on chip (SoCs) that include higher computing capabilities by adding accelerators such as Graphics Processing Units (GPUs) or Tensor Processing

* Corresponding author.
E-mail address: jmorales8@ucam.edu (J. Morales-García).

https://doi.org/10.1016/j.future.2022.12.038
0167-739X/

Table 1
Summary of the main features of GPU virtualization solutions.

Feature	Effect
GPU utilization	Increased
Amount of required GPUs in the cluster	Decreased
The total cost of GPUs in the cluster	Decreased
Energy consumption	Decreased
Execution time for a set of jobs	Decreased

Units (TPUs). Among them, we may highlight Nvidia's Jetson family [6], Intel's branch Movidius [7] or Google's Coral project [8]. Including these accelerators increases the power consumption of edge devices which is already a limiting factor. However, in terms of energy-efficient computing, accelerators substantially reduce application execution time, so that the increased power is amortized. An alternative or additional way to increase the computational capabilities of edge devices without paying for the power increase is the use of remote virtualization of accelerators [9]. This promising mechanism provides hardware resources (mainly GPUs) located on remote nodes to applications in a transparent way to programmers. Moreover, this technique increases GPU utilization thus reducing the waste of energy caused by the idle state of GPUs.

Several GPU virtualization solutions can be found in the literature. Based on CUDA [10], frameworks such as vCUDA [11], DS-CUDA [12], GViM [13], rCUDA [14], GVirtuS [15] or GridCuda [16] provide their own CUDA API implementations that are compatible with the original NVIDIA CUDA library. They are usually based on a client–server approach. The client is a library transparently linked to the CUDA application asking for GPU acceleration on the local node. The server is a daemon running in the remote node that owns the physical GPU. Every time the CUDA application performs a CUDA call, it is caught by the client and forwarded to the server to be executed on the physical GPU. Once the CUDA function is executed, results are returned to the client and forwarded to the CUDA application. It is important to note that the application is not aware of this virtualization process. Moreover, these solutions also allow remote GPUs to be concurrently shared among several applications. Table 1 summarizes the main features of GPU virtualization solutions.

In this work, we introduce a way to increase the computing capabilities of edge devices without increasing their power consumption. To that end, we leverage the rCUDA remote GPU virtualization middleware, using an Nvidia Jetson AGX Xavier as the client node. rCUDA is the best-maintained middleware among the aforementioned frameworks, also offering the best performance compared to other remote GPU virtualization solutions available as well as being available for all processor architectures [17]. The evaluation is carried out with the Fuzzy Minimals (FM) algorithm [18]: a challenging fuzzy clustering algorithm within the umbrella of ML. The main contributions of this paper are the following:

1. We provide a novel way to increase the computing power of edge devices without penalizing their power consumption. Our results show that a 3.2x speed-up factor can be achieved by reducing the power usage by up to 30% using a single remote GPU.
2. A multi-GPU implementation of the FM algorithm is provided to fully leverage multiple GPU instances. Different computational patterns are found in this algorithm, which affects the performance of the remote virtualization framework, concluding that CPU–GPU communication is a critical parameter in this environment.
3. An in-depth evaluation of Nvidia Jetson AGX Xavier is carried out, showing the benefits of introducing accelerators on edge devices for executing heavy workloads.

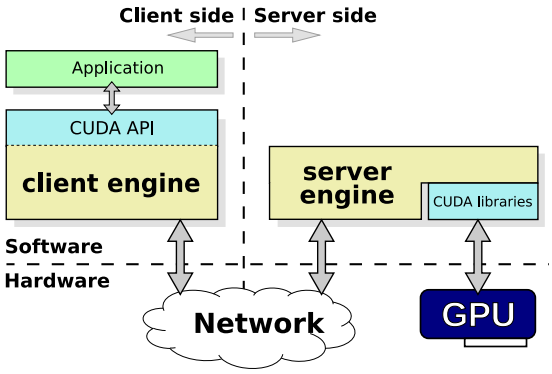


Fig. 1. Architecture of the rCUDA middleware.

4. A thorough evaluation of the rCUDA middleware is performed as an alternative to provide the devices on the edge with increased computing capabilities. Several communication technologies are studied to assess the impact of communications within this framework.

The paper is organized as follows. We briefly introduce some preliminary concepts about the rCUDA middleware and the FM algorithm in Section 2. The parallelization in multi-GPU systems is introduced in Section 3. In Section 4 we describe the experimental results before we conclude the paper with a brief discussion and consideration of future work.

2. Background

2.1. The rCUDA middleware

The architecture of the rCUDA middleware, shown in Fig. 1, follows a client–server distributed approach and works as described in the previous section (intercepting CUDA calls and forwarding them to the remote server). We refer the reader to a full description of rCUDA [19].

It is important to remark that rCUDA works in a completely transparent way to programmers, who do not have to change the source code of their applications in order to leverage this middleware. Actually, if the application is compiled to dynamically use the CUDA library, then the application does not even need to be recompiled to be executed with rCUDA. In this regard, rCUDA is binary compatible with CUDA 9.2 and implements the entire CUDA Runtime and Driver APIs (except for graphics functions). It also provides support for the libraries included within CUDA (cuBLAS, cuFFT, etc.). Additionally, it supports several underlying interconnection technologies by making use of a set of network-specific communication modules (currently TCP/IP, RoCE and InfiniBand). Independently of the exact network used, data exchange between rCUDA clients and servers is pipelined in order to attain high performance. Internal pipeline buffers within rCUDA use preallocated pinned memory, given the higher throughput of this type of memory [20].

rCUDA allows applications to make use of a technique known as multi-tenancy. This mechanism consists of providing several virtual instances of the same remote GPU to the same client application. That is, instead of providing the client application with several remote GPUs, all of them being different physical GPUs, with rCUDA it is possible to partition a physical GPU into several virtual instances and provide them to the same client application. In this way, the client application will have the illusion of being accessing several remote GPUs although all of those remote GPUs are finally mapped onto the same physical GPU. We will make use of multi-tenancy later in this paper.

It is important to notice that rCUDA guarantees the same isolation and security features among virtual GPU instances mapped onto the same physical GPU as the features guaranteed by CUDA. This is achieved by using a different GPU context for each of the virtual GPU instances being run at the same virtual GPU. Therefore, as far as CUDA guarantees isolation among GPU contexts when rCUDA is used, there cannot exist a data leak among applications concurrently using the same physical remote GPU.

2.2. The Fuzzy Minimals algorithm

Clustering analysis consists of assigning data points to clusters (or groups) so that points belonging to the same cluster are as similar as possible, whereas points belonging to another group are as different as possible to the former. This similarity process is based on the evaluation (i.e. minimization) of an objective function, which includes measures such as distance, connectivity, and/or intensity. Different objective functions may be chosen, depending on the dataset features or the application. Fuzzy clustering is a set of classification algorithms where each data point can belong to more than one cluster. Typical examples of these algorithms include the Fuzzy C-Means (FCM) and Fuzzy Minimals (FM) algorithms [21,22]. The latter was initially proposed by Flores-Sintas et al. where authors demonstrate that it meets the expected characteristics of a classification algorithm; i.e. scalability, adaptability, self-driven, stability, and data-independent. In addition, the FM algorithm does not require setting the number of prototypes to be identified in the dataset, as it makes use of k-means or FCM algorithms. In what follows, a brief description of FM is provided. We refer the reader to [22,23] for insights.

Let X be a set of n data points, such that

$$X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^F,$$

where F is the dimension of the vector space.

Algorithm 1 Fuzzy Minimals algorithm, where X is the input dataset to be classified, V is the algorithm output that contains the prototypes found by the clustering process. F is the dimension of the vector space.

1: Choose ε_1 and ε_2 standard parameters.
2: Initialize $V = \{\} \subset \mathbb{R}^F$.
3: *Load_Dataset*(X)
4: $r =$ *Calculate_r_Factor*(X)
5: *Calculate_Prototypes*$(X, r, \varepsilon_1, \varepsilon_2, V)$

Algorithm 1 shows the sequential baseline of the FM algorithm. This algorithm has two main procedures: (1) the calculation of the factor r (line 4) and (2) the calculation of prototypes or centroids (lines 5) to complete the set V. The factor r is a parameter that measures the isotropy in the data set. The use of Euclidean distance implies that the homogeneity and isotropy of the feature space are assumed. Whenever homogeneity and isotropy are broken, clusters are created in the feature space.

$$\frac{\sqrt{|C^{-1}|}}{nr^F} \sum_{x \in X} \frac{1}{1 + r^2 d_{xm}^2} = 1. \tag{1}$$

The calculation of factor r is based on Eq. (1). It is a non-linear expression, where $|C^{-1}|$ is the determinant of the inverse of the covariance matrix, m is the mean of the sample X, d_{xm} is the Euclidean distance between x and m, and n is the number of elements in the sample.

Once the factor r is calculated, the calculation of prototypes is executed to obtain the clustering result in V (see Algorithm 2). The objective function used by FM is given by Eq. (2)

$$J_{(v)} = \sum_{x \in X} \mu_{xv} \cdot d_{xv}^2, \tag{2}$$

Algorithm 2 *Calculate_Prototypes*() of fuzzy minimals algorithm.

1: **for** $k = 1; k < n; k = k + 1$ **do**
2: $v_{(0)} = x_k$, $t = 0$, $E_{(0)} = 1$
3: **while** $E_{(t)} \geq \varepsilon_1$ **do**
4: $t = t + 1$
5: $\mu_{xv} = \frac{1}{1 + r^2 \cdot d_{xv}^2}$, using $v_{(t-1)}$
6: $v_{(t)} = \frac{\sum_{x \in X} \left(\mu_{xv}^{(t)}\right)^2 \cdot x}{\left(\mu_{xv}^{(t)}\right)^2}$
7: $E_{(t)} = \sum_{\alpha=1}^{F} \left(v_{(t)}^{\alpha} - v_{(t-1)}^{\alpha}\right)$
8: **end while**
9: **if** $\sum_{\alpha}^{F} (v^{\alpha} - w^{\alpha}) > \varepsilon_2, \forall w \in V$ **then**
10: $V \equiv V + \{v\}$.
11: **end if**
12: **end for**

where

$$\mu_{xv} = \frac{1}{1 + r^2 \cdot d_{xv}^2}, \tag{3}$$

Eq. (3) measures the degree of membership for a given element x to the cluster where v is the prototype. The FM algorithm is an iterative procedure that aims to minimize the objective function through Eq. (4), giving the prototypes represented by each cluster. Finally, ε_1 and ε_2 are input parameters that establish the error degree committed in the minimum estimation and show the difference between potential minimums respectively.

$$v = \frac{\sum_{x \in X} \mu_{xv}^2 \cdot x}{\sum_{x \in X} \mu_{xv}^2} \tag{4}$$

3. Parallelization strategies for the Fuzzy Minimals on multi-GPU systems

This section presents the CUDA parallelization of the FM algorithm in multi-GPU environments. Parallelization of the FM algorithm on a single GPU was initially proposed in [18]. However, the rCUDA middleware allows applications to use many GPUs on a single node in a shared memory fashion. Thus, the FM algorithm is redesigned to fully leverage this new computational landscape. Our multi-GPU parallelization approach is based on the distributed parallelization of the FM algorithm introduced in Timon et al. [22], called the Parallel Fuzzy Minimals (PFM); a parallel version of the FM algorithm to improve data-parallelism. Contrary to other proposals, the FM algorithm does not require cluster comparison to minimize the objective function. The PFM takes advantage of this feature to split the original data set into different subsets to which the FM is independently applied. The clustering procedure is not affected by this partition as the global properties of the original data set are not lost. Actually, each subset will be classified using an objective function that includes the factor r, which has information about the overall data structure. In fact, the factor r is a global parameter that contains information about the entire data set (see Fig. 2).

In order to benefit from the PFM's intrinsic parallelism, the factor r must be first calculated using the whole dataset (see Fig. 2). As previously explained, the factor r calculation is based on the resolution of a non-linear expression that is computationally intensive. Moreover, the factor r calculation grows exponentially with the number of variables to be clustered [24]. This may become a bottleneck. Therefore, the first goal is to leverage the available multi-GPU system for the factor r calculation. Algorithm 3 shows the multi-GPU implementation of computing factor r. The factor r procedure is based on two main tasks that are applied

Algorithm 3 *Factor r calculation algorithm in GPU.*

```
1: #pragma omp parallel for private (detvalue) schedule (dynamic) reduction (+ : rfactor)
2: for i = 1; i < rows; i++ do
3:    cudaSetDevice(omp_get_thread_num())
4:    covariance <<< bl, th, sh >>> (dataset, determ, row_i, rows, cols)
5:    LU_solver <<< bl, th, sh >>> (determ)
6:    cudaMemcpy(detvalue, determ, sizeof(float), cudaMemcpyDeviceToHost)
7:    rfactor+ = 1/sqrt(detvalue)
8: end for
```

Algorithm 4 *Multi-GPU parallelization of the prototype calculation.*

```
1: #pragma omp parallel for private(jE_total, jE_reduction, mu_total, mu_reduction, mu, prot, prov_prot, aux)
2: for i = 1; i < rows; i++ do
3:    cudaSetDevice(omp_get_thread_num())
4:    cudaMemcpy(prot, (dataset + i * columns), columns * sizeof(FLOAT), cudaMemcpyDeviceToDevice);
5:    while jE_total > error do
6:       new_prot <<< bl, th, sh >>> (dataset, mu, prot, prov_prot);
7:       reduction_mu <<< 1, columns >>> (mu, mu_reduction);
8:       cudaMemcpy(mu_total, mu_reduction, sizeof(FLOAT), cudaMemcpyDeviceToDevice)
9:       distance_prots <<< 1, columns >>> (jE, prov_prot, prot, mu_total, columns);
10:      reduction_jE <<< 1, columns >>> (jE, jE_reduction);
11:      cudaMemcpy(jE_total, jE_reduction, sizeof(FLOAT), cudaMemcpyDeviceToDevice)
12:      aux = prot
13:      prot = prov_prot
14:      prov_prot = aux
15:   end while
16:   cudaMemcpy(prot_provisional + i * columns, prot, columns * sizeof(FLOAT), cudaMemcpyDeviceToHost);
17: end for
```

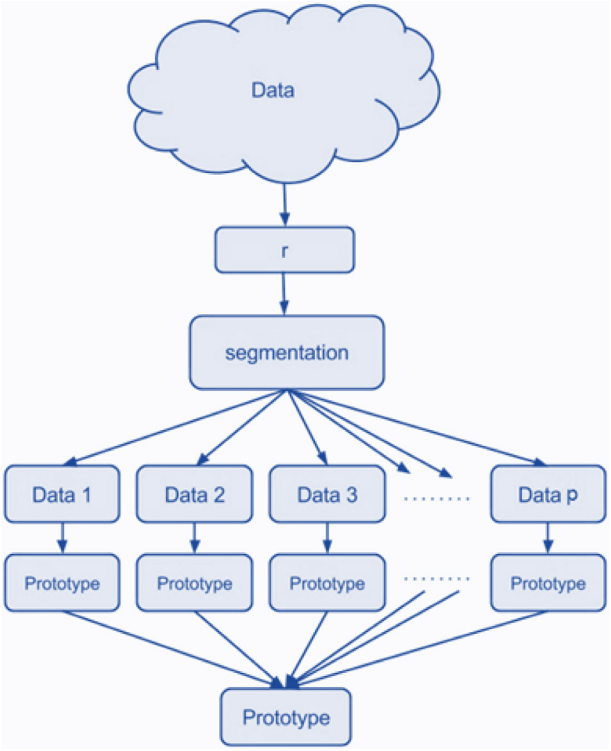


Fig. 2. Parallel implementation of fuzzy minimals clustering algorithm based on [22].

to each row in the dataset. They are (1) the calculation of the fuzzy covariance matrix, which is a square matrix (*columns* × *columns*) and (2) the calculation of its determinant. It is an iterative procedure with as many iterations as rows (points in the datasets). These iterations can be carried out independently and thus they can be equally distributed among the number of GPUs available in the system. This is achieved by parallelizing the for loop with as many CPU threads as GPUs are available in the rCUDA client node (see line 1, algorithm 3). OpenMP is used to do this task and also to eventually reduce the factor r. Moreover, the performance is also affected by the number of columns (i.e. the dimension of points F). Actually, performance will be drastically affected by this last parameter, since the execution time of the determinant calculation, which is carried out in the GPU using the LU method, will increase exponentially with the size of the matrix. The fuzzy covariance matrix is then calculated to obtain factor r.

The multi-GPU parallelization of prototype calculation is shown in Algorithm 4. Here, the input dataset is equally divided into subgroups that are distributed among the GPUs available on the rCUDA client node. As with the factor r parallel design, this is achieved by parallelizing the outermost for loop (see Lines 1–3, Algorithm 4). Each GPU thread calculates the distance between each row of the data set and the different prototypes by calculating Eq. (3) to obtain the probability of belonging to each subset. Finally, in each GPU, a block reduction must be performed to obtain the total degree of relevance for each point in the dataset. It is worth highlighting the large number of synchronizations and data transfers between CPU and GPU this function has.

4. Evaluation and discussion

4.1. Hardware setup and benchmarking

Our experimental environment includes nine nodes (see Fig. 3). Seven out of these nine nodes will be used as GPU servers by leveraging the rCUDA middleware whereas the other two nodes will be used to execute the FM application. Regarding the GPU server nodes, six of them are equipped with processors

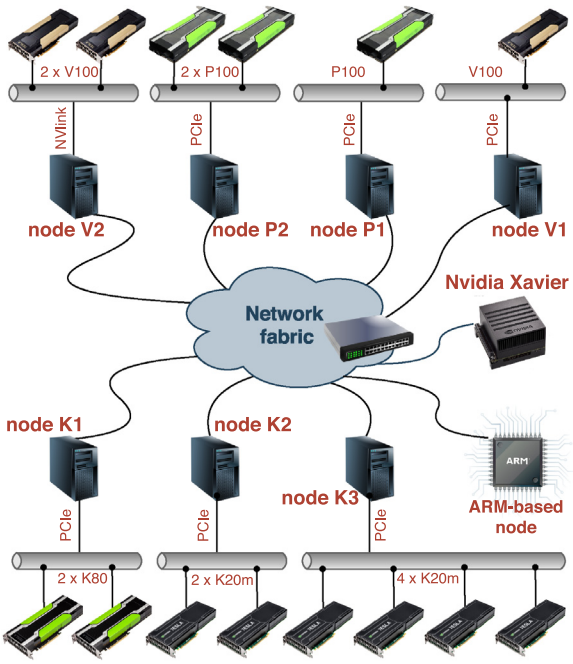


Fig. 3. Hardware environment.

belonging to different Intel Xeon generations. They comprise several Nvidia GPU architectures, ranging from old Kepler (2xK80 and 6xK20) to Pascal (3xP100) and Volta (1xV100) architectures. GPUs in these nodes are connected to the rest of the nodes by a PCI express v3.0 link (16 GB/s). The seventh GPU server is an IBM POWER 9 system with two 16-core Power 9 processors at 2.7 GHz including two Nvidia V100 GPUs. These GPUs are connected to the processors and between them through an NVLink high-speed interconnect with a theoretical bandwidth equal to 300 GB/s. All of these seven nodes in the cluster are connected among them by two network technologies: Gigabit Ethernet (1 Gbps) and EDR InfiniBand (100 Gbps).

In addition to the seven GPU-server nodes, our testbench includes two additional nodes where the FM application will be executed along with the rCUDA client library. The first of these two nodes is an edge computing Nvidia AGX Jetson Xavier. This device is connected to the cluster through Gigabit Ethernet (1Gbps) and has an 8-core ARM v8.2 64-bit CPU, 8 MB L2 + 4 MB L3, 512-core Volta GPU with Tensor Cores, and 16 GB 256-Bit LPDDR4x running at 137 GB/s. It is important to note that this is where the FM computation is performed using either the local GPU or the remote hardware resources provided by rCUDA.

The connection of the Nvidia AGX Jetson Xavier with the GPU servers is one of the main infrastructure bottlenecks. The remote GPU virtualization strategy to bring hardware resources from the GPU servers to the rCUDA client requires intensive communication between the rCUDA client and the remote server processes. Therefore, the technology used in this connection should be evaluated to identify the main bandwidth and latency requirements in this virtualized context. However, Nvidia Jetson Xavier has only an Ethernet interface, which limits this evaluation. Therefore, an additional ARM-based node with two different network interfaces, Gigabit Ethernet and EDR InfiniBand, has been considered. This node has two 28-core ARM processors ThunderX2 at 2 GHz with 256 GB RAM and will be used as an rCUDA client in order to test the connection among the rCUDA clients and GPU servers.

Table 2
GPU models and hardware location of the experimental environment.

GPU ID	Model	Location	GPU ID	Model	Location
0	P100	Node P1	7	K80	Node K1
1	P100	Node P2	8	K20m	Node K2
2	P100	Node P2	9	K20m	Node K2
3	V100	Node V1	10	K20m	Node K3
4	V100	Node V2	12	K20m	Node K3
5	V100	Node V2	12	K20m	Node K3
6	K80	Node K1	13	K20m	Node K3

Table 3
Relationship between experiment tag and GPU ID shown in Table 2. Please refer to Table 2 in order to know the exact GPU location in the cluster.

Tag (amount of GPUs)	Devices used (GPU IDs)	Tag (amount of GPUs)	Devices used (GPU IDs)
1 GPU	0	8 GPUs	0,1,2,3,4,5,6,7
2 GPUs	0,1	10 GPUs	0,1,2,3,4,5,6,7 8,9
3 GPUs	0,1,2	12 GPUs	0,1,2,3,4,5,6,7 8,9,10,11
4 GPUs	0,1,2,3	14 GPUs	0,1,2,3,4,5,6,7 8,9,10,11,12,13
6 GPUs	0,1,2,3,4,5		

Tables 2 and 3 summarize the hardware resources available for the evaluation and the keywords used for the ongoing experimental results.

An interesting discussion is how general is the hardware configuration leveraged in our experiments. In this regard, there are three considerations to be done: (1) is the NVIDIA Jetson Xavier node representative of actual edge nodes? (2) are the nodes hosting the real GPUs representative of current cluster nodes? (3) is the ThunderX2 node representative of current edge devices? Regarding (1), please notice that edge devices are based on ARM processors, as the NVIDIA Jetson Xavier is. However, the latter device comprises a powerful GPU, which could not be a common choice in edge deployments. In this regard, it is important to remark that, in our experiments, when we use remote GPUs according to our proposal, the GPU located within the NVIDIA Jetson Xavier node is not used. Actually, that GPU is only used for comparison purposes, and it is not part of our edge proposal. Therefore, using the NVIDIA Jetson Xavier device in our experiments is representative of current edge deployments. Regarding (2), the nodes we have used to host the real GPUs are a good example of what it could be found in current clusters. Thus, the combination of the NVIDIA Jetson Xavier and the cluster nodes used in this paper represents a general hardware solution (as far as the GPU used in the NVIDIA Jetson Xavier is not used). Finally, regarding (3), the big ThunderX2 ARM node used in this paper is not representative of the current edge devices. However, notice that we only use that node in order to deeper analyze the behavior of our proposal when a faster network is leveraged, as it could be the case when 5G or 6G infrastructure is widely available. In summary, the hardware configuration used in our experiments is general enough to get general conclusions from the experiments.

In order to calculate the energy consumption of our proposal, we have measured, at intervals of one second, the power consumed by each of the devices used. The power consumed by Nvidia Jetson AGX Xavier has been measured using the Watts Up Pro power meter. Regarding the K20, K80, P100 and V100 GPUs, their power consumption has been measured using Nvidia Management Library (NVML).

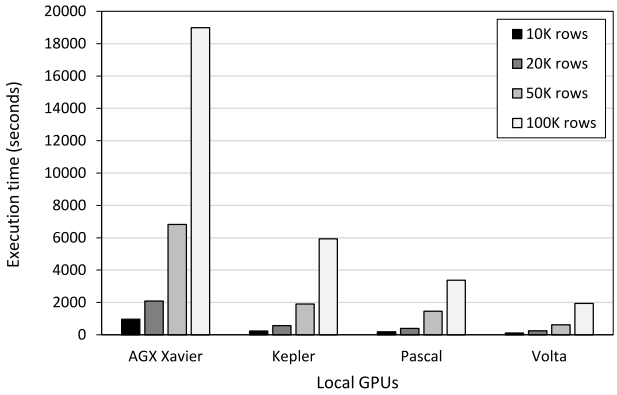


Fig. 4. Execution time of FM algorithm using a local GPU for a different number of rows.

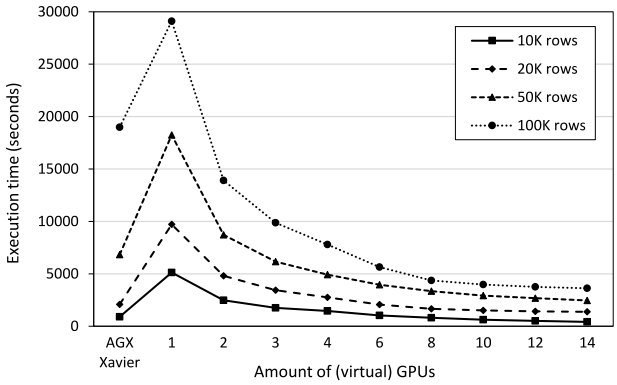


Fig. 5. Execution time of FM algorithm on AGX Xavier, varying the number of virtual GPUs and also the number of rows. Execution times for the AGX Xavier node are used as a reference.

Finally, the dataset to test our implementation is taken from the UCI Machine Learning Repository [25]. It is an input database with up to 100 K points, belonging to different chemical sensors. These sensors have been provided with a set of parameters that measure the quality of the gas, as well as the temperature and humidity values of the environment.

4.2. Performance and energy evaluation

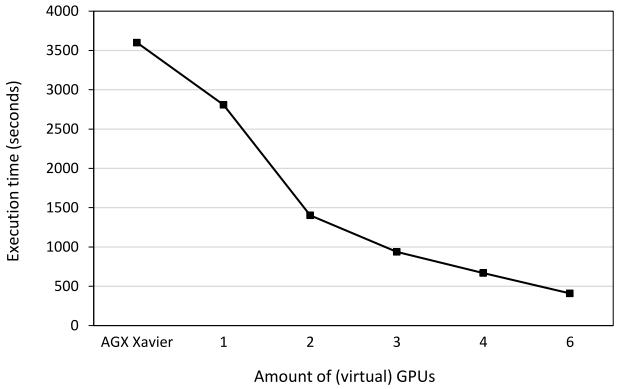
Fig. 4 shows the execution time of the FM algorithm on different GPU architectures. These GPU architectures are those described in Section 4.1, which include the edge computing platform Xavier and Nvidia HPC GPU solutions K80 (Kepler), P100 (Pascal), and V100 (Volta). Only one local GPU has been used in each experiment. The difference in performance between Xavier and HPC GPUs is substantial as expected, reaching up to 13.2X speed-up factor. Computationally speaking, Xavier offers a low-power solution with some computational capabilities but there is still a big gap in performance between HPC GPUs and edge computing platforms and therefore cloud-based approaches are still mandatory to deploy high computational workloads such as those within the umbrella of ML.

Our main goal in this paper is to introduce the computational horsepower available in the cloud into the edge transparently to the programmer. In this way, programmers can develop edge computing-based applications with high-performance capabilities without penalizing power consumption. With this approach, programmers could also use already existing multi-GPU codes that were developed for shared memory multi-GPU nodes on edge computing platforms.

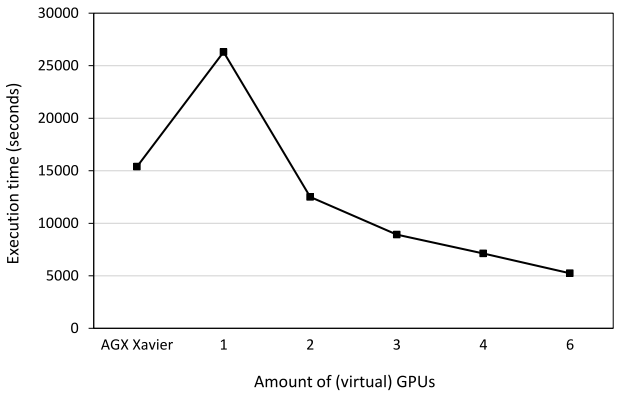
In the search for our goal, Fig. 5 shows the FM execution times on the Xavier while varying the number of virtual GPUs available at the edge platform. In other words, the rCUDA client is executed at the AGX Xavier node where several rCUDA servers are running at different nodes where HPC GPUs are plugged into. It is worth highlighting that only one GPU process is running at each physical remote GPU, which means up to 14 physical GPUs are used in the cloud to provide these performance figures. Table 3 shows the exact GPUs used for each experiment, whereas Table 2 provides the location for each GPU. Fig. 5 shows an important concern: while all virtualized GPUs have higher performance than the Xavier AGX (see Fig. 4) the rCUDA middleware introduces some communication overhead, which can be perfectly seen in Fig. 5 when only one virtual GPU is used, thus worsening the performance of the low-power GPU available on Xavier. However, as more virtualized GPUs are introduced into the edge, the overall application performance gets better. This is particularly true when the data set is large enough because the overhead introduced by rCUDA communications is hidden by the higher computational load assigned to each of the GPUs. This leads us to believe that for small problem sizes, GPUs are not busy enough and, therefore, the computation could be optimized by better using remote GPUs, as will be explained later in this section.

As explained in Section 2.2, the FM algorithm is divided into two main functions: *Calculate_r_Factor* and *Calculate_Prototypes*. They have different computational patterns that affect their performance in the virtualized environment targeted. Fig. 6(a) shows the execution time of the *Calculate_r_Factor* function when this function addresses a large dataset composed of 100,000 rows and up to 11 different variables (columns), which is actually a very challenging scenario. The factor r calculation is made up of several GPU kernels that are executed independently. Actually, there is only a final reduction to get the final factor r. This means that there are few CPU–GPU or GPU–GPU communications in the loop which actually benefits the system's scalability. Remember that CPU–GPU communication in this virtualized environment means having communication over the network which penalizes the performance. Results in Fig. 6(a) show the execution time of the factor r calculation in the local GPU included in the Jetson AGX Xavier and also using up to 6 remote GPUs. It can be seen in the figure that execution time for the factor r calculation progressively decreases along with the number of GPUs, showing almost linear scalability. On the other hand, Fig. 6(b) shows the execution time of the prototype calculation. Here, the results are quite different because the CPU–GPU communication ratio is higher than in the factor r counterpart version. The set of prototypes obtained in each iteration by each GPU is shared with the rest of the GPUs. This means that the increment in network traffic must be compensated by adding more GPUs in the system that allow reducing computation time. This effect happens from 2 remote GPUs and beyond and this pattern is transferred to all the computations since the calculation of prototypes represents 95% of the total computation of the algorithm.

Summing up, there are two main conclusions from this initial analysis. The first conclusion is that remote virtualization of GPUs is drastically penalized by the CPU–GPU communication overhead. This makes sense since CPUs are connected to GPUs via low-latency and high bandwidth connections such as PCI Express 3.0 (approx. 16 GB/s) or NVLINK (approx. 300 GB/s). On the contrary, in rCUDA each CPU–GPU transfer goes through the network fabric, thus depending on the connectivity technology that is used in the cluster. The above results are based on Ethernet, whose bandwidth is around 100 MB/s. This is at least an order of

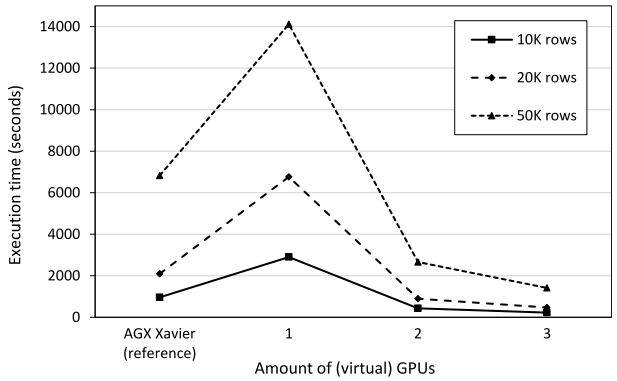


(a) Execution time of factor r

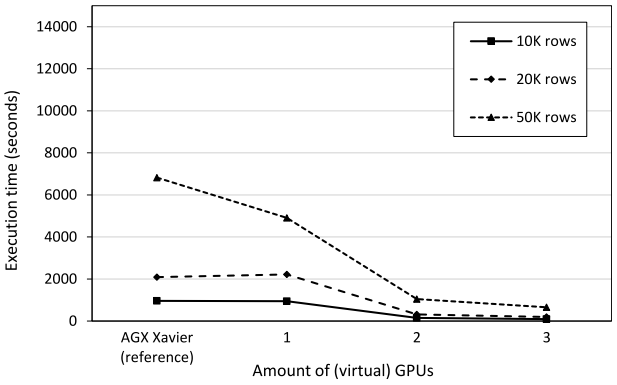


(b) Execution time of prototypes calculation

Fig. 6. Execution time of factor r and prototypes calculation for 100,000 rows. The number of virtual GPUs are varied up to 6 virtual GPUs. The host node is the AGX Xavier that we also use as baseline.



(a) ARM-based node with Ethernet



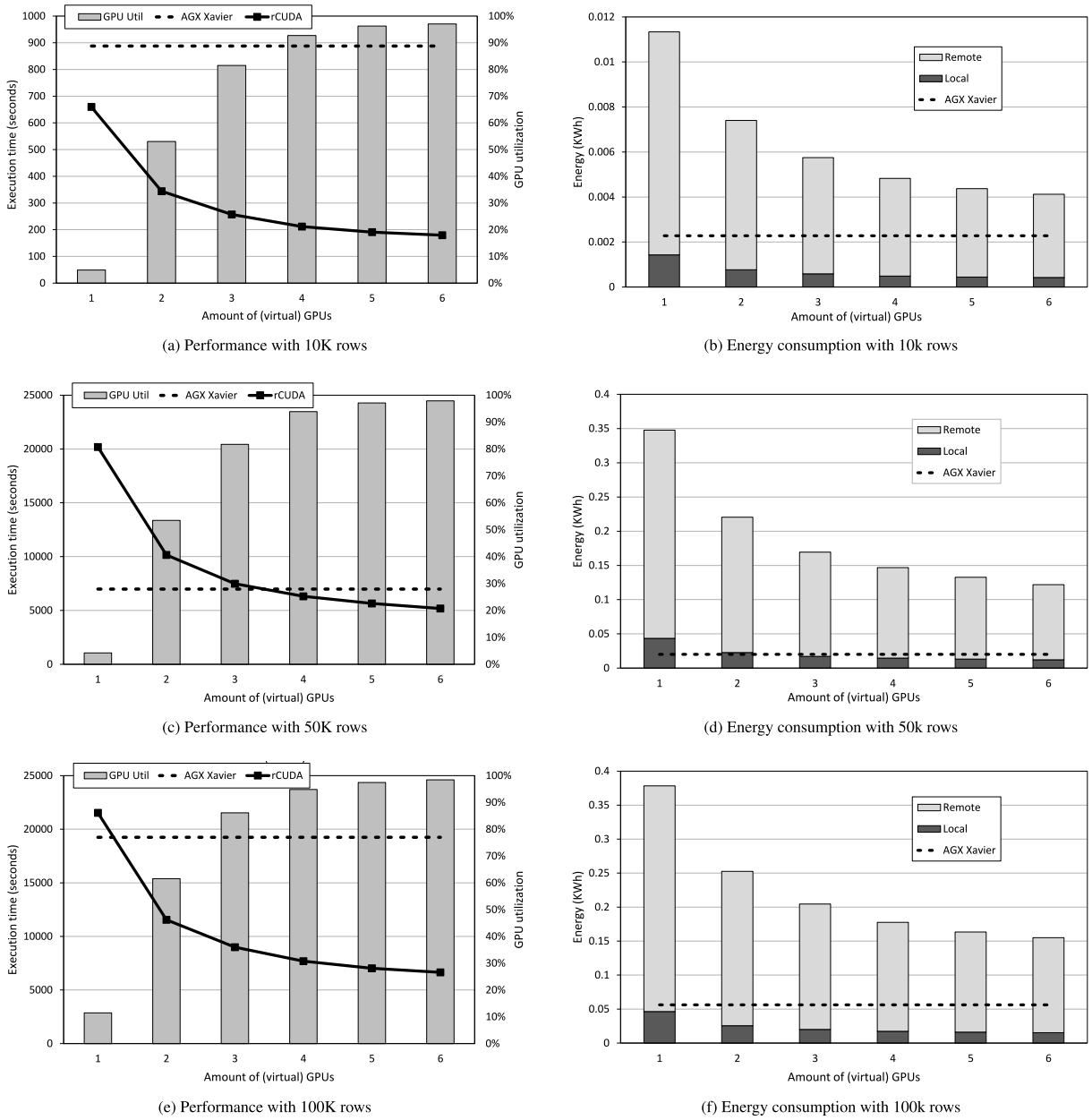
(b) ARM-based node with InfiniBand

Fig. 7. Execution time of FM algorithm, varying the number of virtual GPUs for different number of rows from ARM-based node with Ethernet and InfiniBand.

magnitude less than the scenario within the node. This is actually shown in Fig. 7 where the execution time of the FM algorithm is shown, varying the interconnection technology (i.e. Gigabit Ethernet in Fig. 7(a) and InfiniBand in Fig. 7(b)). It is important to note that experiments in these figures have been carried out in the ARM-based node instead of AGX Xavier as the latter does not have an InfiniBand interface. With the improvement in the interconnection technology, execution time using remote GPUs is dramatically reduced, reaching up to 3x speed-up factor.

The second conclusion from this initial analysis is that it looks like GPUs are not busy enough and therefore we may think about making better use of them. In this regard, in order to increase GPU utilization we may think about multi-tenancy. That is, instead of running one single GPU process per physical GPU as in the previous experiments, we may execute several GPU processes in the same physical GPU. This can be achieved by virtualizing multiple times a single physical GPU and providing multiple virtual instances of that GPU to the FM application.

This multi-tenancy approach is used in Fig. 8, which shows performance and energy consumption when remote GPU virtualization is used in Nvidia Jetson Xavier but only one remote GPU (i.e. V100) is physically targeted although this GPU supports several GPU instances virtually. This increases GPU utilization as it is reflected by the bars of Figs. 8(a), 8(c), and 8(e). Almost 100% of the GPU utilization is achieved with 6 virtual GPUs running on the same V100. Indeed, this is translated into a performance gain as the continuous line shows (the dashed line shows the yield horizon set by Xavier's performance). Almost full utilization of the V100 resources (80%) is required to match Xavier's performance for the medium datasets (i.e. 50 K). As for 10 K and 100 K datasets (Figs. 8(a) and 8(e)), Xavier's performance is defeated with 1–2 Virtual GPUs, achieving a speed-up factor in the range of 3.5X to 4.5X with 6 virtual GPUs within the same physical V100. It is important to note that the performance using remote virtualization techniques does not entirely depend on the data size. Actually, there is another important factor that also penalizes performance as shown in Figs. 7(a) and 7(b); i.e. the number of CPU–GPU communications. This last factor depends on the number of synchronization, cudaMemcpy, CUDA kernel launches, and so on, carried out by the application, which in the case of FM, and clustering algorithms in general, depends on the convergence criterion as a function of the layout of the input data. The number of calls to *cudaMemcpyDeviceToHost* for 50 K is 906,493 and 892,587 for 100 K, which means less CPU–GPU communications for 100 K case. On the right-hand side of Fig. 8, the energy consumption of the FM algorithm is shown. In this case, the dashed line shows Xavier's energy consumption when running the FM algorithm locally; i.e. using its low-power GPU. It can be seen in Figs. 8(b), 8(d) and 8(f) that overall energy consumption is reduced as the multi-tenancy degree increases. Actually, with 6 virtual instances, overall energy consumption is less than one-half of the energy required with a single remote virtual instance. On the other hand, overall energy consumption is always larger than when the FM algorithm is executed locally in



(a) Performance with 10K rows

(b) Energy consumption with 10k rows

(c) Performance with 50K rows

(d) Energy consumption with 50k rows

(e) Performance with 100K rows

(f) Energy consumption with 100k rows

Fig. 8. Execution time and energy consumption of the FM algorithm at local AGX Xavier and varying the number of virtual GPUs within the same physical GPU (V100).

Nvidia Xavier edge node. However, it can be seen that the energy required by the edge node noticeably decreases when several virtual instances are employed. This is better analyzed in Fig. 9 where the average power consumption of the edge computing device and NVIDIA V100, running the FM algorithm for 10 K, 50 K, and 100 K is shown.

Fig. 9 shows that the average power consumption of this edge device when the GPU is active is 10.15 W with 0.5 standard deviations. Furthermore, Xavier's power consumption is reduced by a factor of up to 30% when the execution is offloaded to a remote GPU. Xavier's local GPU can be switched off in this scenario, reducing its power budget in the range of 7–8 W. As the number of virtual GPU instances increases, Xavier's CPU workload also increases, and this is translated into slightly higher power consumption. Actually, the power consumption for 1 virtual GPU is 7.77 W and increases up to 8.3 for 6 virtual GPUs. But even in this worst-case scenario, our results show 21.15% power savings over using Xavier's local GPU. The server-side power consumption also depends on the workload supported by Nvidia V100 used in our experiments. It ranges from 54.54 W for 1 GPU instance to 75.80 W for 6 GPUs instances. The power consumption of V100 is higher than Xavier (up to 7.5x) but it offers better performance as expected (up to 3.1x speed-up factor using 6 GPU instances). A trade-off between power consumption and performance is clearly reported; the edge/fog computing devices such as Nvidia Jetson Xavier are low-power devices because they are designed to be in IoT infrastructures where power consumption could become a big issue. By delegating the heaviest workloads to the GPU servers, up to 30% of the power consumption could be transferred to a more suitable and supportive infrastructure. In addition, performance can be gained but taking into account that the total power consumption of the solution is penalized. Summing up, it is important to note that the execution time is reduced in exchange for increasing both the power and energy consumed in the IoT infrastructure as a whole (i.e. GPU servers and rCUDA client).

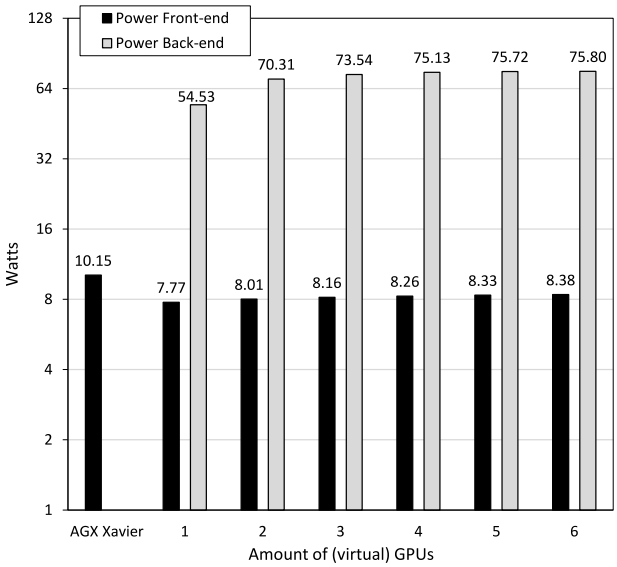


Fig. 9. Average power consumption of Nvidia AGX Xavier (rCUDA client) and Nvidia V100 (GPU server), running 10 K, 50 K and 100 K datasets and varying the number of virtual GPUs within the same physical GPU.

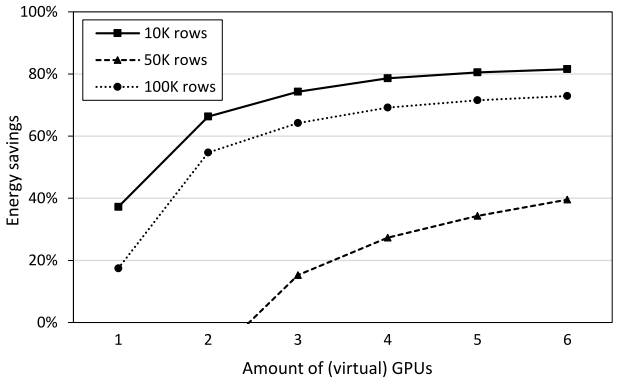


Fig. 10. Energy savings achieved at the Jetson AGX Xavier node when using remote virtual GPUs with respect to local executions. 10 K, 50 K, and 100 K datasets are run while varying the number of virtual GPUs within the same physical GPU.

The most remarkable result to emerge from the data, however, is that the rCUDA client is where there are usually problems with energy and power supply in IoT infrastructures, and our approach reduces the execution time, power, and energy at this level of the architecture. Actually, Fig. 10 clearly shows the energy savings achieved at the rCUDA client (Jetson AGX Xavier) thanks to the usage of a remote virtualized GPU providing multiple instances of virtual GPUs. In order to compute energy savings, the energy required by the AGX Xavier node when executing the FM algorithm with the local small GPU was used as the baseline. It can be seen in the figure that, the more virtual GPU instances are leveraged, the larger energy savings are attained. Energy savings, when 6 remote virtual GPU instances are used, ranging from 40% for 50 K rows up to 81% for 10 K rows. It is noteworthy that in the case of 100 K rows energy savings are about 70%. These extraordinary energy savings could translate into much longer battery availability, for instance. On the other hand, the previous discussions about the behavior of the FM algorithm for 50 K rows also apply to this figure, where it can be seen that energy savings for this particular amount of rows are noticeably lower than for the other amounts of rows as the performance gains of using remote virtualization is lower than in the other case. But even with these lower performance benefits, the energy savings reach up to 40%.

5. Conclusions and future work

This paper evaluates the virtualization of remote GPUs in edge computing devices in order to provide them with computational horsepower without increasing the power consumption of these devices. A particular case study is developed with a fuzzy clustering algorithm (Fuzzy Minimals, FM) which is widely used for different data science applications. The FM's GPU version has two main kernels with different CPU–GPU communication patterns that strongly affect this kind of GPU virtualization. The factor r calculation shows little CPU–GPU communication while the calculation of the prototypes needs continuous global synchronization in each step of the algorithm. Our results show that new edge computing platforms, which include low-power GPUs such as Nvidia Jetson Xavier, provide an excellent framework for driving edge computing as a real alternative to smart applications. However, they have yet to further improve their performance to meet the challenge of the data deluge in IoT ecosystems. Remote virtualization of GPUs can be a good solution to increase computing power without increasing the power budget of edge devices. We report a performance gain of up to 3.1x speed-up factor by just using a single remote GPU running multiple virtual GPU instances on it. Moreover, the power consumption of the edge device is reduced by up to 30%, obtaining up to 80% of energy savings by delegating the GPU workload to the rCUDA servers as its GPU can remain off. However, the overall power consumption of the IoT infrastructure (rCUDA clients and remote GPU servers) is increased by a factor of 7x with the solution presented here. Anyway, notice that the GPU servers are not a dedicated infrastructure but they can provide service to multiple applications. Therefore, the increase in overall energy consumption is diluted, making the energy saving in the edge device more appealing.

The conjunction of virtualization and edge computing is still at a relatively early stage; we emphasize that we have only so far tested a relatively simple variant of this solution that is designed for supercomputer environments. We definitely think that designing low-power virtualization solutions can reduce overall power consumption by maintaining the performance gains for edge devices. Moreover, there are many other types of data science algorithms still to explore, and as such, it is a potentially fruitful area of research. However, the different kernels of the FM algorithm have different computational and communication patterns that provide interesting conclusions about the computational patterns that can benefit from this edge computing virtualized infrastructure. This and other algorithms could also be tested on frameworks other than the one tested in this paper in order to assess the impact of edge device characteristics on the performance and power consumption of these algorithms. We hope that this paper stimulates further discussion and work.

Finally, another direction for future work is to model performance vs. power consumption, varying the different hardware parameters used in this work, in order to provide the research community with a useful tool to evaluate the benefits of using remote GPU virtualization in the context of edge computing.

CRediT authorship contribution statement

José M. Cecilia: Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Juan Morales-García:** Software, Formal analysis. **Baldomero**

Imbernón: Software, Formal analysis. **Javier Prades:** Software, Formal analysis, Visualization. **Juan-Carlos Cano:** Methodology, Validation, Supervision, Project administration, Funding acquisition. **Federico Silla:** Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jose M. Cecilia, Baldomero Imbernón, Juan-Carlos Cano and Federico Silla report financial support was provided by Spanish Ministry of Science and European Commission.

Data availability

Data will be made available on request.

Acknowledgment

All authors have read and agreed to the published version of the manuscript.

Funding

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017861 and also by projects RTC2019-007159-5 and Ramon y Cajal Grant RYC2018-025580-I, funded by MCIN/AEI/10.13039/501100011033, "FSE invest in your future" and "ERDF A way of making Europe". The authors are grateful for the generous support provided by Mellanox Technologies Inc.

Consent to publish

All authors have read and agreed to the published version of the manuscript.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660.
[2] M. Duranton, K. De Bosschere, C. Gamrat, J. Maebe, H. Munk, O. Zendra, The HiPEAC vision 2017, 2017.
[3] K. Papadokostaki, G. Mastorakis, S. Panagiotakis, C.X. Mavromoustakis, C. Dobre, J.M. Batalla, Handling big data in the era of internet of things (IoT), in: Advances in Mobile Cloud Computing and Big Data in the 5G Era, Springer, 2017, pp. 3–22.
[4] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39.
[5] M. Capra, R. Peloso, G. Masera, M. Ruo Roch, M. Martina, Edge computing: A survey on the hardware requirements in the internet of things world, Future Internet 11 (4) (2019) 100.
[6] M. Ditty, T. Architecture, J. Montrym, C. Wittenbrink, NVIDIA's Tegra K1 system-on-chip, in: 2014 IEEE Hot Chips 26 Symposium (HCS), IEEE, 2014, pp. 1–26.
[7] Movidius, Movidius: On-Device computer vision & AI, URL: https://www.movidius.com/.
[8] Coral, Coral: Build intelligent ideas with our platform for local AI, URL: https://coral.withgoogle.com/.
[9] B. Imbernón, J. Prades, D. Gimenez, J.M. Cecilia, F. Silla, Enhancing large-scale docking simulation on heterogeneous systems: An MPI vs rCUDA study, Future Gener. Comput. Syst. 79 (2018) 26–37.
[10] C. Nvidia, CUDA C programming guide, version 10.1, NVIDIA Corp, 2019.
[11] L. Shi, H. Chen, J. Sun, K. Li, vCUDA: GPU-accelerated high-performance computing in virtual machines, IEEE Trans. Comput. 61 (6) (2011) 804–816.
[12] M. Oikawa, A. Kawai, K. Nomura, K. Yasuoka, K. Yoshikawa, T. Narumi, DS-CUDA: a middleware to use many GPUs in the cloud environment, in: 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, IEEE, 2012, pp. 1207–1214.
[13] V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, P. Ranganathan, GViM: GPU-accelerated virtual machines, in: Proceedings of the 3rd ACM Workshop on System-Level Virtualization for High Performance Computing, 2009, pp. 17–24.
[14] F. Silla, S. Iserte, C. Reaño, J. Prades, On the benefits of the remote GPU virtualization mechanism: The rCUDA case, Concurr. Comput.: Pract. Exper. 29 (13) (2017).
[15] G. Giunta, R. Montella, G. Agrillo, G. Coviello, A GPGPU transparent virtualization component for high performance computing clouds, in: European Conference on Parallel Processing, Springer, 2010, pp. 379–391.
[16] T.-Y. Liang, Y.-W. Chang, GridCuda: a grid-enabled CUDA programming toolkit, in: 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications, IEEE, 2011, pp. 141–146.
[17] F. Silla, J. Prades, E. Baydal, C. Reaño, Improving the performance of physics applications in atom-based clusters with rCUDA, J. Parallel Distrib. Comput. 137 (2020) 160–178.
[18] J.M. Cebrian, B. Imbernón, J. Soto, J.M. García, J.M. Cecilia, High-throughput fuzzy clustering on heterogeneous architectures, Future Gener. Comput. Syst. 106 (2020) 401–411.
[19] J. Prades, F. Silla, GPU-job migration: The rCUDA case, IEEE Trans. Parallel Distrib. Syst. 30 (12) (2019) 2718–2729.
[20] C. Reaño, F. Silla, Redesigning the rCUDA communication layer for a better adaptation to the underlying hardware, Concurr. Comput.: Pract. Exper. (2020).
[21] J. Bezdek, R. Ehrlich, W. y Full, FCM: The fuzzy C-means clustering algorithm, Comput. Geosci. 10 (2–3) (1984) 191–203.
[22] I. Timón, J. Soto, H. Pérez-Sánchez, J.M. Cecilia, Parallel implementation of fuzzy minimals clustering algorithm, Expert Syst. Appl. 48 (2016) 35–41.
[23] J. Soto, A. Flores-Sintas, J. y Palarea-Albaladejo, Improving probabilities in a fuzzy clustering partition, Fuzzy Sets and Systems 159 (4) (2008) 406–421.
[24] J.M. Cecilia, I. Timón, J. Soto, J. Santa, F. Pereñíguez, A. Muñoz, High-throughput infrastructure for advanced ITS services: A case study on air pollution monitoring, IEEE Trans. Intell. Transp. Syst. 19 (7) (2018) 2246–2257.
[25] D. Dua, C. Graff, UCI machine learning repository, 2017, URL: https://archive.ics.uci.edu/ml/datasets/Gas+sensors+for+home+activity+monitoring.

José M. Cecilia is a Ramón y Cajal research fellow (Associate Professor Tenure track) at the Computer Engineering Department, Universitat Politècnica de València (Spain). He received his B.S. degree in Computer Science from Universidad de Murcia (Spain, 2005), M.S. degree in Computational Software Techniques in Engineering from Cranfield University (United Kingdom, 2007), and Ph.D. degree in Computer Science from the Universidad de Murcia (Spain, 2011). Dr. Cecilia is a well-established researcher in the high-performance computing (HPC) area, specifically for the applications he has developed. His publication portfolio is composed of 50 research papers in JCR-Ranked journals, and 37 contributions on high-quality international conferences in the area (including ParCo, IJCNN, ICPP or GTC). Dr. Cecilia's research career has been focused on bridging the gaps between hardware and software by developing novel applications that overcome current technological barriers. Particularly, he has developed applications within the convergence between emergent HPC and Artificial Intelligence, following an application-driven approach where hardware and algorithmic innovations have been always considered to offer novel solutions to real problems. More information: http://www.grc.upv.es/members.html.

Juan Morales-García is a Professor , Researcher and Ph.D. student at the Computer Science Department, Catholic University of Murcia (Spain). He received his B.Sc. and M.Sc. degrees in Computer Science from the Catholic University of Murcia in 2020 and 2021, respectively. His current research interests include Machine Learning, Deep Learning and Pervasive computing.

Baldomero Imbernón is an assistant professor at the Catholic University of Murcia (Spain). He received his BS degree in computer science from Catholic University of Murcia (Spain, 2013) and MS degree in new technologies in computer science from the University of Murcia (Spain, 2015) specialized in high performance architectures and supercomputing. He received his Ph.D. in computer science at Catholic University of Murcia in 2018. He has co-authored over 30 publications among JCR-indexed journals and national and international conferences. His research interests include bioinformatics, high performance computing and optimization algorithms in GPU.

Javier Prades received the M.S. degree in Computer Engineering from Universitat Politècnica de València, Spain, in 2015. He is currently pursuing his Ph.D. studies in Computer Science at that university. His areas of interest include heterogeneous architectures, GPGPU computing, HPC, high performance interconnection networks, virtualization solutions, and cloud computing.

Juan-Carlos Cano, SMIEEE, is a full professor in the Department of Computer Engineering at the Polytechnic University of Valencia (UPV) in Spain. He earned an M.Sc. and a Ph.D. in Computer Science from the UPV in 1994 and 2002 respectively. From 1995–1997 he worked as a programming analyst at IBM's manufacturing division in Valencia. His current research interests include Wireless Communications, Vehicular Networks, Mobile Ad Hoc Networks, and Pervasive Computing.

Federico Silla received the MS and Ph.D. degrees in Computer Engineering from Universitat Politècnica de València, Spain, in 1995 and 1999, respectively. He is currently Full Professor at the Department of Computer Engineering at that university. Furthermore, he worked for two years at Intel Corporation developing on-chip networks. His research addresses high performance on-chip and off-chip interconnection networks as well as remote GPU virtualization mechanisms. He has published 30 JCR indexed journal papers, more than 100 contributions to international conferences, 10 book chapters, 43 invited talks and 12 tutorials, providing an H-index impact factor equal to 28 according to Google Scholar. He is also associate editor of the JPDC journal as well as the coordinator of the rCUDA remote GPU virtualization project since it began in 2008. More information: http://www.disca.upv.es/fsilla.

CAPÍTULO III –

RESULTADOS

CAPÍTULO III – RESULTADOS

A continuación, se expone un resumen y discusión de los resultados obtenidos a lo largo del desarrollo de esta tesis doctoral, las conclusiones extraídas y las posibles futuras líneas de investigación de este trabajo.

3.1. Resumen y discusión de los resultados obtenidos

En el primer artículo se propone una recogida de datos proveniente de dispositivos IoT con su correspondiente preprocesamiento así como el desarrollo y evaluación de técnicas de ML. En el seno de esta investigación se recogen los datos en bruto (generando el dataset inicial -*RAW*-), se preprocesan para eliminar valores anómalos, outliers, valores fuera de rango, etc. (generando el dataset sanitizado -*CLEAN*-) y, posteriormente se suavizan mediante técnicas conocidas como los filtros de Kalman (generando el dataset suavizado -*SMOOTH*-). Una vez obtenidos y preprocesados los datos, se implementan las técnicas de ML, en concreto, se han implementado cuatro distintas: AutorRegressive (AR), AutoRegressive Integrated Moving Average (ARIMA), K-Nearest Neighbors Regressor (KNNR) y Random Forest Regressor (RFR). Para cada modelo implementado, se ha evaluado la calidad de la predicción de cada uno de los datasets generados.

En el segundo artículo se propone el desarrollo y evaluación de técnicas de DL así como una metodología para reducir la dimensionalidad de dichas técnicas para que puedan ser ejecutadas en plataformas de bajas capacidades de cómputo tales como los dispositivos IoT. En el seno de esta investigación se estudia la adaptación y ejecución de diversas técnicas de AI (CNN y LSTM) en varios dispositivos IoT (Arduino, Raspberry, Nvidia Jetson Nano y Nvidia Jetson Xavier). El estudio evalúa diversos datasets (*RAW*, *CLEAN* y *SMOOTH*) con distintas granularidades (15 minutal, 30 minutal y 60 minutal) obteniendo, para cada modelo, sus métricas de calidad y consumo energético.

En el tercer artículo se propone una arquitectura descentralizada que permite entrenar y predecir con modelos predictivos de AI en entornos en tiempo real. En el seno de esta investigación se construye una arquitectura que es capaz de recibir datos de sensores, limpiar los datos recibidos, almacenarlos en base de datos, entrenar modelos predictivos de AI (como las redes neuronales MLP, CNN, LSTM y CNNLSTM), realizar predicciones precisas en tiempo real, establecer los periodos de re-entrenamiento y la comunicación entre los servidores HPC y dispositivos IoT. En concreto, se estudian los tiempos de entrenamiento y de predicción de técnicas de AI en servidores de HPC y, por otra parte, se evalúa el tiempo de carga en memoria y de predicción de las mismas técnicas de AI en dispositivos IoT. También se realiza una evaluación exhaustiva de la necesidad de re-entrenamiento de las técnicas de AI conforme avanza el horizonte temporal.

En el cuarto artículo se propone una técnica para aumentar las capacidades de

cómputo de los dispositivos IoT mediante el uso de GPUs remotas virtualizadas sobre dicho dispositivo. En el seno de esta investigación se estudia una técnica de AI conocida como "*Fuzzy Minimals*" (FM) la cual se ha utilizado para uno de los procesos de limpieza de datos: la detección de valores anómalos (*outliers*) dentro de las series temporales. Este algoritmo tiene un alto coste computacional debido a, entre otras cosas, su escasa optimización para el aprovechamiento de los recursos computacionales ofrecidos por las GPUs. Por ello, se ha optimizado el algoritmo previamente mencionado para que pueda aprovechar al máximo los beneficios de la ejecución sobre GPU y, además, se ha ejecutado en dispositivos IoT mediante la utilización de GPUs remotas virtualizadas sobre los mismos. En concreto, se hace uso de distintos tamaños de un conjunto de datos para evaluar la aceleración del algoritmo y la reducción de consumo energético en función del número de GPUs virtualizadas sobre el dispositivo IoT.

3.2. Conclusiones

Tras el análisis de los resultados obtenidos, en el primer artículo presentado en el compendio de publicaciones se han obtenido las siguientes conclusiones:

- En el contexto de invernaderos de agricultura de precisión es necesario implementar sistemas de monitorización de las diferentes variables implicadas en el control climático y de fertirrigación, por ejemplo, dispositivos IoT como sensores para saber qué está ocurriendo en dichas variables en tiempo real.
- El preprocesamiento de los datos afecta a la calidad de la predicción positiva, aunque no significativamente. Es decir, preprocesar los datos mejora ligeramente la calidad de la predicción de los modelos predictivos de ML.
- Los modelos predictivos de ML pueden ser suficientes para predecir lo que va a ocurrir dentro de un invernadero inteligente, aunque la calidad de dicha predicción se puede mejorar mediante el uso de técnicas de DL.

En el segundo artículo presentado en el compendio de publicaciones se han obtenido las siguientes conclusiones:

- El *Edge Computing* lidera la intersección HPC-AI en el ámbito del IoT. Sin embargo, la brecha computacional entre ambas disciplinas sigue siendo enorme. Se necesitan algoritmos más sencillos que ofrezcan soluciones novedosas a las aplicaciones emergentes para hacerlas viables en este contexto, es decir, cumpliendo los requisitos tanto de tiempo de ejecución como de consumo energético.
- El equilibrio entre la complejidad algorítmica, la precisión de los resultados obtenidos y la eficiencia energética es esencial para obtener sistemas robustos y operativos en entornos reales.

- Las técnicas de AI más sencillas pueden ser interesantes en dispositivos IoT sin GPUs integradas, ya que ofrecen un consumo de energía casi un orden de magnitud inferior.
- Los dispositivos IoT con GPU integrada en estos contextos de energía limitada pueden no ser suficientemente fructíferas, a menos que la ejecución del modelo sea lo suficientemente compleja como para necesitarlos.

En el tercer artículo presentado en el compendio de publicaciones se han obtenido las siguientes conclusiones:

- La agricultura de los últimos años está ganando mucho rendimiento cuando las nuevas tecnologías aportan sus ventajas. Prueba de ello es la automatización que se está produciendo en los invernaderos.
- Los dispositivos IoT y los nuevos protocolos de comunicaciones ayudan a conectar sensores con sistemas de recogida de información y toma de decisiones de forma rápida y eficaz, consiguiendo una monitorización y ejecución de acciones rápida y eficiente.
- La arquitectura desarrollada permite al agricultor actuar con antelación para disponer de las mejores condiciones climáticas para sus cultivos, con un modelo fiable y de rápida ejecución en el *edge*.
- Los sistemas de publicación / subscripción son, hoy en día, obligatorios para las aplicaciones IoT que requieren tomar acciones en tiempo real.
- Es necesario contar con una infraestructura IoT estrechamente acoplada y sin fisuras que cubra todo el ciclo de recogida, análisis, procesamiento de datos y predicción de variables cuando se utiliza un generador de datos en tiempo real para monitorizar un ecosistema, siendo necesario tomar decisiones en tiempo real basadas en el análisis de dichos datos y sus previsiones a futuro.

Para finalizar, en el cuarto artículo presentado en el compendio de publicaciones se han obtenido las siguientes conclusiones:

- El uso de técnicas de virtualización de GPUs remotas en dispositivos IoT incrementa de las capacidades de cómputo de los dispositivos IoT.
- El uso de técnicas de virtualización de GPUs remotas en dispositivos IoT decrementa del consumo energético de los propios dispositivos IoT.
- El consumo de energía total de la infraestructura IoT, cuando se hace uso de técnicas de virtualización de GPUs remotas, aumenta notoriamente.

En relación a los objetivos, se puede concluir que se han alcanzado satisfactoriamente todos los objetivos propuestos mediante los artículos científicos que conforman esta tesis doctoral. En concreto:

- El primer objetivo científico se ha cumplido gracias a la aportación realizada en el primer, segundo, tercer y cuarto artículo presentado en esta tesis doctoral. El primer artículo propone la recogida, análisis y transformación de los datos recogidos de pequeños sensores. El segundo artículo propone una adaptación de modelos predictivos de AI para que puedan ser ejecutados en dispositivos IoT con limitadas capacidades computacionales. El tercer artículo propone una arquitectura que permite el despliegue de modelos predictivos de AI en dispositivos IoT. El cuarto artículo propone aumentar las capacidades de cómputo de los dispositivos IoT mediante la virtualización de GPUs remotas para que puedan ser capaces de ejecutar modelos predictivos de AI. En resumen, el proceso de obtención, análisis, procesamiento de datos así como la adaptación de modelos predictivos de AI para reducir sus necesidades computacionales y el incremento de las capacidades computacionales de los dispositivos IoT implican la consecución del primer objetivo científico.

- El segundo objetivo científico se ha cumplido gracias a la aportación realizada en el primer, segundo y tercer artículo presentado en esta tesis doctoral. El primer artículo propone distintos modelos predictivos de ML para realizar predicciones sobre los datos recogidos de los dispositivos IoT. El segundo artículo propone modelos predictivos de DL adaptados a dispositivos con limitadas capacidades computacionales. El tercer artículo propone el despliegue de algoritmos de DL sin reducir sus necesidades computacionales gracias a la arquitectura propuesta. Actualmente, se están desarrollando dos artículos científicos enfocados, el primero, al desarrollo de modelos predictivos de DL multivariantes que contemplan el invernadero como un sistema complejo en el que influyen multitud de variables y, el segundo, al desarrollo algoritmos genéticos que permitan variar las entradas de un modelo de DL multivariante para alcanzar un objetivo específico. En resumen, el desarrollo de modelos predictivos de ML y de DL con el fin de predecir qué es lo que va a suceder en las próximas horas dentro del invernadero con el fin de poder automatizar y optimizar los procesos intrínsecos al invernadero implican la consecución del segundo objetivo científico.

- El objetivo tecnológico se ha cumplido gracias a la aportación realizada en el tercer artículo presentado en esta tesis doctoral. En este artículo se propone una arquitectura IoT que permite desplegar algoritmos de AI que requieren de grandes capacidades de cómputo, solventando las limitaciones inherentes a los dispositivos IoT. Indicar que se ha creado la tecnología necesaria para poder poner en producción todas las investigaciones llevadas a cabo en el seno de esta tesis doc-

toral, creando un prototipo funcional en TRL 3-4 desplegado sobre un escenario real y siendo evaluado actualmente.

Por tanto, el logro de los dos objetivos científicos y el objetivo tecnológico implican la consecución del objetivo general propuesto en esta tesis doctoral.

3.3. Futuras líneas de investigación

La investigación realizada en el seno de esta tesis doctoral abre la puerta a interesantes y novedosas futuras líneas de investigación, detalladas a continuación:

- Estudiar, diseñar e implementar otras técnicas de AI distintas a las ya estudiadas en esta investigación que mejoren la calidad de las predicciones y reduzcan los tiempos de entrenamiento, predicción y el consumo energético.
- Evaluar otros dispositivos IoT en los que desplegar modelos predictivos de AI, que mejoren el rendimiento y minimicen el consumo energético.
- Implementar un sistema de decisión en base a las predicciones obtenidas por los modelos predictivos de AI que sea capaz de realizar acciones de forma autónoma en función de lo que vaya a suceder a corto-medio plazo.
- Descubrir nuevas formas de reducción de las necesidades computacionales de los modelos predictivos de AI para que puedan ser ejecutados en dispositivos IoT minimizando la pérdida de calidad que conlleva la reducción de dimensionalidad de los modelos predictivos.
- Explorar otras tecnologías alternativas distintas a las ya propuestas en la arquitectura de despliegue de modelos predictivos de AI en tiempo real, con el fin de mejorar los tiempos de entrenamiento, carga, predicción y toma de decisiones.
- Implementar otras técnicas de virtualización de GPUs remotas para continuar mejorando las capacidades de cómputo de los dispositivos IoT, reduciendo la latencia y el consumo energético de las comunicaciones que implica el uso de dichas técnicas.
- Transferir todo el conocimiento obtenido en el seno de esta tesis doctoral a otros ámbitos distintos a los de la agricultura intensiva en invernaderos que también necesiten automatizar y optimizar sus procesos para reducir su impacto medioambiental, tales como las técnicas de depuración de aguas residuales, entre otros posibles contextos.

CAPÍTULO IV –

REFERENCIAS BIBLIOGRÁFICAS

CAPÍTULO IV – REFERENCIAS BIBLIOGRÁFICAS

[1] Somayya Madakam et al. «Internet of Things (IoT): A literature review». En: *Journal of Computer and Communications* 3.05 (2015), pág. 164.

[2] Jayavardhana Gubbi et al. «Internet of Things (IoT): A vision, architectural elements, and future directions». En: *Future generation computer systems* 29.7 (2013), págs. 1645-1660.

[3] Ling Qian et al. «Cloud computing: An overview». En: *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*. Springer. 2009, págs. 626-631.

[4] Jianli Pan y James McElhannon. «Future edge cloud and edge computing for internet of things applications». En: *IEEE Internet of Things Journal* 5.1 (2017), págs. 439-449.

[5] Ang Li et al. «CloudCmp: comparing public cloud providers». En: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 2010, págs. 1-14.

[6] Keyan Cao et al. «An overview on edge computing research». En: *IEEE access* 8 (2020), págs. 85714-85728.

[7] Mahadev Satyanarayanan et al. «The case for vm-based cloudlets in mobile computing». En: *IEEE pervasive Computing* 8.4 (2009), págs. 14-23.

[8] Tolga Soyata et al. «Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture». En: *2012 IEEE symposium on computers and communications (ISCC)*. IEEE. 2012, págs. 000059-000066.

[9] Md Fazlay Rabbi Masum Billah y Muhammad Abdullah Adnan. «Smartlet: a dynamic architecture for real time face recognition in smartphone using cloudlets and cloud». En: *Big Data Research* 17 (2019), págs. 45-55.

[10] Pieter Simoens et al. «Scalable crowd-sourcing of video from mobile devices». En: *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. 2013, págs. 139-152.

[11] Athanasios Voulodimos et al. «Deep learning for computer vision: A brief review». En: *Computational intelligence and neuroscience* 2018 (2018).

[12] SK Lakshmanaprabu et al. «Optimal deep learning model for classification of lung cancer on CT images». En: *Future Generation Computer Systems* 92 (2019), págs. 374-382.

[13] Cyril Voyant et al. «Machine learning methods for solar radiation forecasting: A review». En: *Renewable energy* 105 (2017), págs. 569-582.

[14] A Belayneh et al. «Coupling machine learning methods with wavelet transforms and the bootstrap and boosting ensemble approaches for drought prediction». En: *Atmospheric research* 172 (2016), págs. 37-47.

[15] Ashwani Kumar Tiwari et al. «Hydrogeochemical analysis and evaluation of surface water quality of Pratapgarh district, Uttar Pradesh, India». En: *Applied Water Science* 7 (2017), págs. 1609-1623.

[16] Konstantinos G Liakos et al. «Machine learning in agriculture: A review». En: *Sensors* 18.8 (2018), pág. 2674.

[17] Rami Al-Rfou et al. «Theano: A Python framework for fast computation of mathematical expressions». En: *arXiv e-prints* (2016), arXiv-1605.

[18] Yangqing Jia et al. «Caffe: Convolutional architecture for fast feature embedding». En: *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, págs. 675-678.

[19] Adam Paszke et al. «Pytorch: An imperative style, high-performance deep learning library». En: *Advances in neural information processing systems* 32 (2019).

[20] Tianqi Chen et al. «Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems». En: *arXiv preprint arXiv:1512.01274* (2015).

[21] Martín Abadi et al. «Tensorflow: a system for large-scale machine learning.» En: *Osdi*. Vol. 16. 2016. Savannah, GA, USA. 2016, págs. 265-283.

[22] Burak Saltuk. «Current situation in Mediterranean greenhouses and a structural analysis example (Mersin province)». En: (2018).

[23] T Soriano et al. «A study of direct solar radiation transmission in asymmetrical multi-span greenhouses using scale models and simulation models». En: *Biosystems Engineering* 88.2 (2004), págs. 243-253.

[24] Selmin Burak y Jean Margat. «Water management in the Mediterranean region: concepts and policies». En: *Water Resources Management* 30 (2016), págs. 5779-5797.

[25] Peter J Batt. *Managing Agricultural Value Chains in a Rapidly Urbanizing World*. 2022.

[26] Zhiling Ren et al. «Managing energy-water-carbon-food nexus for cleaner agricultural greenhouse production: A control system approach». En: *Science of The Total Environment* 848 (2022), pág. 157756.

[27] Zhaoyu Zhai et al. «Decision support systems for agriculture 4.0: Survey and challenges». En: *Computers and Electronics in Agriculture* 170 (2020), pág. 105256.

[28] Menghang Zhang et al. «Energy-saving design and control strategy towards modern sustainable greenhouse: A review». En: *Renewable and Sustainable Energy Reviews* 164 (2022), pág. 112602.

[29] Anil Bhujel et al. «Sensor systems for greenhouse microclimate monitoring and control: a review». En: *Journal of Biosystems Engineering* 45 (2020), págs. 341-361.

[30] João P Romero y Camila Gramkow. «Economic complexity and greenhouse gas emissions». En: *World Development* 139 (2021), pág. 105317.

[31] Natasja Ariesen-Verschuur, Cor Verdouw y Bedir Tekinerdogan. «Digital Twins in greenhouse horticulture: A review». En: *Computers and Electronics in Agriculture* 199 (2022), pág. 107183.

[32] LIU Dan et al. «Intelligent agriculture greenhouse environment monitoring system based on IOT technology». En: *2015 International Conference on Intelligent Transportation, Big Data and Smart City*. IEEE. 2015, págs. 487-490.

[33] S Revathi, Thota K Radhakrishnan y N Sivakumaran. «Climate control in greenhouse using intelligent control algorithms». En: *2017 American Control Conference (ACC)*. IEEE. 2017, págs. 887-892.

[34] Morteza Taki et al. «Applied machine learning in greenhouse simulation; new application and analysis». En: *Information processing in agriculture* 5.2 (2018), págs. 253-268.

[35] Cesar Hernandez et al. «Modeling of energy demand of a high-tech greenhouse in warm climate based on bayesian networks». En: *Mathematical Problems in Engineering* 2015 (2015).

[36] Zhili Chen et al. «A water-saving irrigation decision-making model for greenhouse tomatoes based on genetic optimization TS fuzzy neural network». En: *KSII Transactions on Internet and Information Systems (TIIS)* 13.6 (2019), págs. 2925-2948.

[37] Taleb Zarei y Reza Behyad. «Predicting the water production of a solar seawater greenhouse desalination unit using multi-layer perceptron model». En: *Solar Energy* 177 (2019), págs. 595-603.

[38] Andreas Kamilaris y Francesc X Prenafeta-Boldú. «Deep learning in agriculture: A survey». En: *Computers and electronics in agriculture* 147 (2018), págs. 70-90.

[39] Yang Lu et al. «Identification of rice diseases using deep convolutional neural networks». En: *Neurocomputing* 267 (2017), págs. 378-384.

[40] Harshana Habaragamuwa et al. «Detecting greenhouse strawberries (mature and immature), using deep convolutional neural network». En: *Engineering in Agriculture, Environment and Food* 11.3 (2018), págs. 127-138.

[41] Albert Reuther et al. «Survey and benchmarking of machine learning accelerators». En: *2019 IEEE high performance extreme computing conference (HPEC)*. IEEE. 2019, págs. 1-9.

[42] Sparsh Mittal. «A survey of FPGA-based accelerators for convolutional neural networks». En: *Neural computing and applications* 32.4 (2020), págs. 1109-1139.

[43] Yu Emma Wang, Gu-Yeon Wei y David Brooks. «Benchmarking TPU, GPU, and CPU platforms for deep learning». En: *arXiv preprint arXiv:1907.10701* (2019).

[44] Yuxin Wang et al. «Benchmarking the performance and energy efficiency of AI accelerators for AI training». En: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE. 2020, págs. 744-751.

[45] Firas Al-Ali et al. «Novel casestudy and benchmarking of AlexNet for edge AI: From CPU and GPU to FPGA». En: *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE. 2020, págs. 1-4.

[46] Raju Machupalli, Masum Hossain y Mrinal Mandal. «Review of ASIC accelerators for deep neural network». En: *Microprocessors and Microsystems* 89 (2022), pág. 104441.

[47] Umair Saeed et al. «A Review of Structural Testing Methods for ASIC based AI Accelerators». En: *IJCSNS* 23.1 (2023), pág. 103.

[48] Haklin Kimm, Incheon Paik y Hanke Kimm. «Performance Comparision of TPU, GPU, CPU on Google Colaboratory Over Distributed Deep Learning». En: *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE. 2021, págs. 312-319.

[49] Stefano Markidis et al. «Nvidia tensor core programmability, performance & precision». En: *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE. 2018, págs. 522-531.

[50] Kim Hazelwood et al. «Applied machine learning at facebook: A datacenter infrastructure perspective». En: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE. 2018, págs. 620-629.

[51] Sanjaya K Panda y Prasanta K Jana. «An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems». En: *Cluster Computing* 22.2 (2019), págs. 509-527.

[52] Mehboob Hussain et al. «Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing». En: *Sustainable Computing: Informatics and Systems* 30 (2021), pág. 100517.

[53] Guibin Wang y Xiaoguang Ren. «Power-efficient work distribution method for cpu-gpu heterogeneous system». En: *International symposium on parallel and distributed processing with applications*. IEEE. 2010, págs. 122-129.

[54] Xiaoyong Tang y Zhuojun Fu. «CPU–GPU utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems». En: *IEEE Access* 8 (2020), págs. 58948-58958.

[55] Sylvain Collange, David Defour y Arnaud Tisserand. «Power consumption of GPUs from a software perspective». En: *Computational Science–ICCS 2009: 9th International Conference Baton Rouge, LA, USA, May 25-27, 2009 Proceedings, Part I 9*. Springer. 2009, págs. 914-923.

[56] Mohammad Sadrosadati et al. «ITAP: Idle-time-aware power management for GPU execution units». En: *ACM Transactions on Architecture and Code Optimization (TACO)* 16.1 (2019), págs. 1-26.

[57] Susanta Nanda Tzi-cker Chiueh y Stony Brook. «A survey on virtualization technologies». En: *Rpe Report* 142 (2005).

[58] Belen Bermejo, Carlos Juiz y Carlos Guerrero. «Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance». En: *The Journal of Supercomputing* 75.2 (2019), págs. 808-836.

[59] Shailaja Salagrama y Vimal Bibhu. «Study of IT and Data Center Virtualization». En: *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*. Vol. 2. IEEE. 2022, págs. 274-278.

[60] Giulio Giunta et al. «A GPGPU transparent virtualization component for high performance computing clouds». En: *Euro-Par 2010-Parallel Processing: 16th International Euro-Par Conference, Ischia, Italy, August 31-September 3, 2010, Proceedings, Part I 16*. Springer. 2010, págs. 379-391.

[61] Carlos Reaño et al. «Local and remote GPUs perform similar with EDR 100G InfiniBand». En: *Proceedings of the Industrial Track of the 16th International Middleware Conference*. 2015, págs. 1-7.

[62] Baldomero Imbernon et al. «Enhancing large-scale docking simulation on heterogeneous systems: An MPI vs rCUDA study». En: *Future Generation Computer Systems* 79 (2018), págs. 26-37.

[63] Farzad Samie, Lars Bauer y Jörg Henkel. «From cloud down to things: An overview of machine learning in internet of things». En: *IEEE Internet of Things Journal* 6.3 (2019), págs. 4921-4934.

CAPÍTULO V –

ANEXOS

CAPÍTULO V – ANEXOS

5.1. Datos relativos a la calidad de las publicaciones

A continuación, se exponen los datos relativos a la calidad de las publicaciones presentadas en esta tesis doctoral.

5.1.1. Data-driven evaluation of machine learning models for climate control in operational smart greenhouses (Journal of Ambient Intelligence and Smart Environments)

La tabla 5 muestra los datos relativos a la calidad de la revista en la que se publicó el primer artículo (ver tabla 1) perteneciente al compendio de publicaciones.

Datos relativos a la calidad de la revista	
Título	Journal of Ambient Intelligence and Smart Environments
ISSN	1876-1364
eISSN	1876-1372
Editorial	IOS Press
Factor de impacto (JCR)	2.759
Indicador de citas (JCI)	0.48
Categoría	Computer Science, Artificial Intelligence
Ranking	84/145 (Q3)
Página web	https://www.iospress.com/catalog/journals/journal-of-ambient-intelligence-and-smart-environments

Tabla 5: Datos relativos a la calidad de la revista en la que se publicó el primer artículo del compendio de publicaciones.

5.1.2. Evaluation of low-power devices for smart greenhouse development (Journal of Supercomputing)

La tabla 6 muestra los datos relativos a la calidad de la revista en la que se publicó el segundo artículo (ver tabla 2) perteneciente al compendio de publicaciones.

Datos relativos a la calidad de la revista	
Título	Journal of Supercomputing
ISSN	0920-8542
eISSN	1573-0484
Editorial	Springer
Factor de impacto (JCR)	2.557
Indicador de citas (JCI)	0.77
Categoría	Computer Science, Theory & Methods
Ranking	43/110 (Q2)
Página web	https://www.springer.com/journal/11227

Tabla 6: Datos relativos a la calidad de la revista en la que se publicó el segundo artículo del compendio de publicaciones.

5.1.3. *SEPARATE: A tightly coupled, seamless IoT infrastructure for deploying AI algorithms in smart agriculture environments (Internet of Things)*

La tabla 7 muestra los datos relativos a la calidad de la revista en la que se publicó el tercer artículo (ver tabla 3) perteneciente al compendio de publicaciones.

Datos relativos a la calidad de la revista	
Título	Internet of Things
ISSN	2543-1536
eISSN	2542-6605
Editorial	Elsevier
Factor de impacto (JCR)	5.711
Indicador de citas (JCI)	1.27
Categoría	Computer science, Information systems
Ranking	31/164 (Q1)
Página web	https://www.sciencedirect.com/journal/internet-of-things

Tabla 7: Datos relativos a la calidad de la revista en la que se publicó el tercer artículo del compendio de publicaciones.

5.1.4. *Using remote GPU virtualization techniques to enhance edge computing devices (Future Generation Computer Systems)*

La tabla 8 muestra los datos relativos a la calidad de la revista en la que se publicó el cuarto artículo (ver tabla 4) perteneciente al compendio de publicaciones.

Datos relativos a la calidad de la revista	
Título	Future Generation Computer Systems
ISSN	0167-739X
eISSN	1872-7115
Editorial	Elsevier
Factor de impacto (JCR)	7.307
Indicador de citas (JCI)	2.39
Categoría	Computer Science, Theory & Methods
Ranking	10/110 (Q1)
Página web	https://www.sciencedirect.com/journal/future-generation-computer-systems

Tabla 8: Datos relativos a la calidad de la revista en la que se publicó el cuarto artículo del compendio de publicaciones.

5.2. Otras publicaciones científicas

A continuación, se muestran otras publicaciones científicas fruto de la colaboración con investigadores del grupo de investigación y del proyecto GLOBALoT.

5.2.1. *Revistas científicas*

Los artículos publicados en revistas científicas (y que no forman parte del compendio de publicaciones de esta tesis doctoral) son los siguientes:

- Cecilia, J. M., Cano, J. C., **Morales-García, J.**, Llanes, A., & Imbernón, B. (2020). Evaluation of clustering algorithms on GPU-based edge computing platforms. *Sensors*, *20*(21), 6335. Q1. https://doi.org/10.3390/s20216335.
- Guillén-Navarro, M. A., Martínez-España, R., Bueno-Crespo, A., **Morales-García, J.**, Ayuso, B., & Cecilia, J. M. (2020). A decision support system for water optimization in anti-frost techniques by sprinklers. *Sensors*, *20*(24), 7129. Q1. https://doi.org/10.3390/s20247129.

- García-Cremades, S., **Morales-García, J.**, Hernández-Sanjaime, R., Martínez-España, R., Bueno-Crespo, A., Hernández-Orallo, E., López-Espín, J. J., & Cecilia, J. M. (2021). Improving prediction of COVID-19 evolution by fusing epidemiological and mobility data. *Scientific Reports, 11*(1), 1-16. Q2. https://doi.org/10.1038/s41598-021-94696-2.

Además de los artículos científicos previamente mencionados, se han presentado a revistas científicas los siguientes artículos, los cuales se encuentran en proceso de revisión:

- **Morales-García, J.**, Bueno-Crespo, A., Terroso-Sáenz, F., Arcas-Túnez, F., Martínez-España, R. & Cecilia, J. M. Evaluation of synthetic data generation for intelligent climate control in greenhouses. *Applied Intelligence*. Q2.

- **Morales-García, J.**, Terroso-Sáenz, F., Bueno-Crespo, A., & Cecilia, J. M. An Analysis of Synthetic Timeseries as an Enabler to Improve Region-based Human Mobility Forecasting. *Machine Learning*. Q2.

Para finalizar, también indicar que, actualmente, se están desarrollando los siguientes artículos científicos:

- Implementación y evaluación de modelos de Deep Learning multivariantes en el contexto de invernaderos inteligentes.

- Implementación de un sistema de ayuda a la decisión y optimización de procesos mediante algoritmos genéticos.

- Evaluación de la mejora de las capacidades computacionales de dispositivos IoT mediante rCUDA en un compendio de algoritmos de Inteligencia Artificial.

- Implementación y evaluación de un sistema de alertas en tiempo real en base a las predicciones de los datos recogidos por sensores IoT un invernadero inteligente.

- Predicción de variables meteorológicas en base a unas determinadas coordenadas GPS.

- Predicción de la contaminación acústica y ambiental en España mediante el análisis de datos recogidos de dispositivos IoT.

5.2.2. *Congresos internacionales*

Los artículos presentados a congresos internacionales (y que no forman parte del compendio de publicaciones de esta tesis doctoral) son los siguientes:

- Alati, M. F., Fortino, G., **Morales, J.**, Cecilia, J. M., & Manzoni, P. (2022, January). Time series analysis for temperature forecasting using TinyML. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)* (pp. 691-694). IEEE. Core B.
  https://doi.org/10.1109/CCNC49033.2022.9700573.

- Ruiz, S., **Morales-García, J.**, Calafate, C. T., Cano, J. C., Manzoni, P., & Cecilia, J. M. (2022, June). Evaluation of time-series libraries for temperature prediction in smart greenhouses. In *2022 18th International Conference on Intelligent Environments (IE)* (pp. 1-7). IEEE. Core B.
  https://doi.org/10.1109/IE54923.2022.9826765.

- Ortiz-González, A., Bueno-Crespo, A., Martínez-España, R., Martínez-Más, J., Martínez-Cendán, J. P., Remezal-Solano, M., **Morales-García, J.**, Imbernón, B., Romero-Humber, O., Ortiz-Reina, S., Capozzi, C., & Blasco-Muñoz, S. (2023, May). Ayuda al diagnóstico de la patología cervical mediante segmentación de imágenes de células procedentes de citologías cervicovaginales. In *2023 XXXI Congreso Nacional SEAP-IAP*. Core -.

- Padilla-Quimbiulco, D., **Morales-García, J.**, Cantabella, M., Ayuso, B., Muñoz, A., & Cecilia, J. M. (2023, June). Greenhouse intelligent warning system for precision agriculture. In *2023 19th International Conference on Intelligent Environments (IE)*. IEEE. Core B.

5.2.3. *Workshops*

Los artículos presentados en *workshops* (y que no forman parte del compendio de publicaciones de esta tesis doctoral) son los siguientes:

- **Morales-García, J.**, Llanes, A., Tudela, B. I., & Cecilia, J. M. (2020). Performance Evaluation of Clustering Algorithms on GPUs. In *Intelligent Environments 2020 Workshop Proceedings of the 16th International Conference on Intelligent Environments* (pp. 400-409). IOS Press.
  https://doi.org/10.3233/AISE200066.

- **Morales-García, J.**, Bueno-Crespo, A., & Cecilia, J. M. (2022). Aplicación de Inteligencia Artificial sobre infraestructuras IoT para automatizar y optimizar los

procesos de agricultura intensiva. In *VIII Jornadas de Investigación y Doctorado of the Universidad Católica de Murcia (UCAM)*.

– Arratia, B., Alati, M. F., **Morales-García, J.**, Cecilia, J. M., Calafate, C. T., & Manzoni, P. (2022). Pronóstico de series temporales para datos de temperatura con TinyML. In *Jornadas SARTECO 2022 (Sociedad de Arquitectura y Tecnología de Computadores) of the Universidad de Alicante (UAL)*.

– **Morales-García, J.**, Llanes, A., Curado, M., & Arcas-Túnez, F. (2023). An Exploratory Data Analysis for League of Legends professional match data. In *Intelligent Environments 2023 Workshop Proceedings of the 19th International Conference on Intelligent Environments*. IOS Press.

5.2.4. Innovación docente

Los artículos aceptados en revistas de innovación docente (y que no forman parte del compendio de publicaciones de esta tesis doctoral) son los siguientes:

– **Morales-García, J.**, Llanes, A., Muñoz, A., & Cecilia, J. M. (2023). Un proyecto innovador para acercar a estudiantes de educación superior a la realidad laboral. *Innovación educativa*. Q3.

5.3. TRANSFERENCIA TECNOLÓGICA

Debido al marcado carácter tecnológico de esta investigación, se ha transferido el conocimiento obtenido creando un prototipo funcional en TRL 3-4 que está siendo desplegado en un entorno real, un invernadero inteligente de agricultura de precisión. El prototipo está operativo, obteniendo datos reales del invernadero y ofreciendo recomendaciones para la gestión climática y de fertirrigación de dicho invernadero. Esperamos que pronto se pueda considerar que el producto se encuentra en TRL 5, demostrando la valía de lo investigado en esta tesis doctoral; es decir es posible automatizar y optimizar procesos mediante AIoT para aumentar la producción y reducir el impacto medioambiental.

5.4. PROYECTOS DE INVESTIGACIÓN

Esta tesis doctoral ha sido desarrollada en el contexto del proyecto de investigación "Desarrollo de infraestructuras IoT de altas prestaciones contra el cambio climático basadas en Inteligencia Artificial" (GLOBALoT) con referencia RTC2019-007159-5, financiado por el Ministerio de Ciencia e Innovación / Agencia Estatal de Investigación.

Además, se ha participado en otros proyectos de investigación tales como "Planificación y gestión de recursos hídricos a partir de análisis de datos IoT" (WATERoT), "Sistema inteligente multimodal basado en crowdsensing para un servicio de predicción de problemas sociales" (ALLEGRO) y "Optimización y sostenibilidad de los cultivos intensivos bajo invernadero mediante técnicas de inteligencia artificial e internet de las cosas" (GREENAIoT).

5.5. Colaboraciones con otras entidades

Las colaboraciones (con otras universidades y empresas) consolidadas durante el desarrollo de esta tesis doctoral se muestran a continuación:

- Universidad Politécnica de Valencia (UPV): Se ha establecido una colaboración con el Departamento de Informática de Sistemas y Computadores (DISCA) de dicha universidad, concretamente, con los grupos de investigación *Grupo de Arquitecturas Paralelas (GAP)* y *Grupo de Redes de Computadores (GRC)* pertenecientes a este departamento.
- Nutricontrol S. L.: Se ha establecido una colaboración con dicha empresa para desarrollar, a partir de las investigaciones realizadas en esta tesis doctoral, un producto comerciable utilizado para automatizar y optimizar los procesos presentes en un invernadero de agricultura intensiva.

Estas colaboraciones se pueden ver reflejadas en las publicaciones científicas presentadas en esta tesis doctoral.

5.6. Premios

Los premios obtenidos durante el desarrollo de esta tesis doctoral se muestran a continuación:

- "3 Minutos tesis (3MT)": Finalista de la Universidad Católica de Murcia (UCAM)
- "II Edición del Concurso Tu Tesis Doctoral en un Hilo de Twitter: #HiloTesis": Ganador de la Universidad Católica de Murcia (UCAM) y finalista a nivel nacional.