MAESTRÍA INGENIERÍA COMPUTACIONAL

ARQUITECTURA COMPUTACIONAL DE ANALÍTICA DE DATOS IoT PARA EL CONNECTED HOME BASADA EN DEEP LEARNING

Ing. Carlos Andres Castañeda Osorio
Univesidad de Caldas
Facultad de Ingenierías, Departamento de Sistemas
Manizales
2021

MAESTRÍA INGENIERÍA COMPUTACIONAL

ARQUITECTURA COMPUTACIONAL DE ANALÍTICA DE DATOS IoT PARA EL CONNECTED HOME BASADA EN DEEP LEARNING

Ing. Carlos Andres Castañeda Osorio

Director
Ph.D. Luis Fernando Castillo Ossa

Codirector
Ph.D. Gustavo Adolfo Isaza Echeverry

Univesidad de Caldas
Facultad de Ingenierías, Departamento de Sistemas
Manizales
2021

Resumen

Cada día con mayor frecuencia nos encontramos con dispositivos de uso común conectados a internet que a cada momento capturan datos de su entorno y en muchas ocasiones estos no son procesados o simplemente son almacenados sin ningún sentido práctico. En este proyecto se construyó una arquitectura computacional que permite el análisis de los datos, la transmisión a servicios de nube usando el protocolo *MQTT* y el almacenamiento de los datos en bases de datos no relacionales que faciliten el análisis histórico de los mismos.

El objetivo de la arquitectura es permitir la transmisión segura de los datos y realizar el análisis de los mismos en los dispositivos *IoT* y en servicios de proveedores de la nube usando técnicas de descubrimiento de información como *deep learning*. Para lograr el objetivo propuesto se concluye que es necesario solucionar tres preguntas específicas que son: ¿Cómo transmito los datos?, ¿Cómo y donde analizo los datos? y ¿Cómo almaceno los datos procesados?

Para la transmisión de los datos se encontró que el protocolo *MQTT* permite transmisión eficiente y segura entre dispositivos IoT y también desde los dispositivos hacia servicios de nube usando el protocolo *TLS*. En cuanto al análisis de la información se realizó en dos puntos de la arquitectura, primero en un dispositivo *IoT* conocido como *Raspberry PI* en el cual se pudo analizar datos como imágenes y texto plano usando técnicas como el *deep learning* aplicado con redes neuronales convolucionales; segundo en un proveedor de servicios en la nube aplicando técnicas como máquinas de soporte vectorial *SVM* en un servidor con sistema operativo Linux. Finalmente para el almacenamiento de la información se usó un motor de base de datos no relacional que gracias a su flexibilidad permitió almacenar datos de los resultados de las mediciones directas y de los análisis en los distintos puntos de la arquitectura.

La arquitectura se validó sobre la necesidad presentada por la empresa Mabe, quien en alianza con la Universidad de Caldas y la Universidad Nacional de Colombia permitieron desarrollar el proyecto interinstitucional con código hermes 36715 titulado “Prototipo computacional para la fusión y análisis de grandes volúmenes de datos en entornos *IoT* (Internet de las Cosas) a partir de técnicas de *Machine Learning* y arquitecturas seguras entre sensores, para caracterizar el comportamiento e interacción de los usuarios en un ecosistema de

Connected Home" del cual tenemos como uno de los resultados la validación del presente trabajo.

Agradecimientos

A mi familia por ser parte indispensable dentro de todos los procesos que decido emprender en mi vida recibiendo en cada acción su apoyo incondicional.

A mis directores de tesis PhD. Luis Fernando Castillo Ossa y PhD. Gustavo Adolfo Isaza Echeverry por su conocimiento, disposición y guía durante todo el proceso de construcción del presente trabajo.

A la Universidad de Caldas que como alma mater ha sido testigo de mis logros académicos y me ha permitido desarrollarme y formarme como profesional.

A todas las personas involucradas en el proyecto interinstitucional con código hermes 36715 titulado “Prototipo computacional para la fusión y análisis de grandes volúmenes de datos en entornos *IoT* (Internet de las Cosas) a partir de técnicas de *Machine Learning* y arquitecturas seguras entre sensores, para caracterizar el comportamiento e interacción de los usuarios en un ecosistema de *Connected Home*” que con sus aportes me ayudaron a llevar a buen término el presente trabajo.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Campo Temático

Este trabajo se encuentra enmarcado en el campo del internet de las cosas (*IoT*), el análisis de datos a través de la técnica de *deep learning* aplicado en el *fog computing*, la transmisión segura de información usando protocolos de comunicación como *MQTT* y el uso de servicios de nube como complemento al análisis y almacenamiento de la información.

1.2. Planteamiento del Problema

El mayor objetivo de *IoT* es propiciar que la interacción con el entorno sea ágil, accesible, intuitivo, para ello se le debe facilitar la información que necesita, a través de datos en tiempo real y datos históricos y aplicar inteligencia computacional en la información para tomar decisiones "inteligentes" de manera automática. Esto mejorará nuestra habilidad de manejar nuestras ciudades, carreteras, salud, zonas naturales y mucho más [Chen et al., 2014b, Stolpe, 2016]. Tendremos mejor manejo de multitudes y de tráfico, predicciones de emergencias, mejor predicciones de accidentes y crímenes entre otros. Los datos recolectados de los dispositivos de *IoT* serán usados para entender y controlar ambientes complejos a nuestro alrededor, habilitando una mejor toma de decisiones, mayor automatización, mayor eficiencia, productividad, precisión, y generación de riqueza. El mayor reto en estas configuraciones es el análisis oportuno de grandes cantidades de datos para producir un panorama y unas decisiones altamente fiables y oportunas lo que lograría que *IoT* cumpliese con su promesa. El *Deep Learning* está referenciado como uno de los mejores métodos para extraer conocimiento a partir de los datos de *IoT* [Alam et al., 2016].

El internet de las cosas producirá grandes volúmenes de datos [Bin et al., 2010, Bifet et al., 2010], cada dispositivo conectado generará datos de sus mediciones

relevantes para sí y para los dispositivos de su entorno siendo de vital importancia el análisis de los datos generados, la velocidad de medición y el nivel de control que se defina para un entorno específico, las aplicaciones generalmente se encuentran basados en bases de datos de alto rendimiento o en datos *NoSQL* . Dicho volumen de datos es difícil de analizar dada la pluralidad de fuentes y formatos almacenados, cada dispositivo genera datos de acuerdo a sus intereses y programación propia ocultando así información relevante para otro dispositivo, una persona o incluso una organización.

Aunque existe un consenso sobre la gran importancia del análisis de datos grandes en IoT, se obtienen resultados limitados, especialmente en los fundamentos matemáticos [Bin et al., 2010, Ding et al., 2013, Stolpe, 2016] Por ejemplo, los datos de tráfico son recolectados de muchos dispositivos que colaboran, incluyendo cámaras pre - Implementadas, vehículos conductores y pasajeros, los cuales son generalmente ruidosos, corruptos, heterogéneos, de muchas dimensiones y no son linealmente separables. Para explotar el valor de los datos, el desarrollo de algoritmos efectivos en analítica de datos masivos se necesita. [Li and Yu, 2011, Wu et al., 2014].

Machine learning corresponde a una forma de procesar los datos, para producir información o conocimiento, una de sus técnicas más prevalentes en la actualidad es la denominada *deep learning* o aprendizaje en profundidad, esta se encuentra entre los enfoques más prometedores para lograr el desafío de extraer inferencias precisas de datos de sensores sin procesar [Lane et al., 2015], sin embargo se ha identificado que actualmente los algoritmos de *deep learning* están iniciando en *IoT* y que aunque funcione en muchos casos no se ha realizado un estudio detallado para estimar bajo cuales condiciones funciona de la mejor manera, motivo por lo cual es un aspecto clave en este proyecto determinar las características de aplicación de este técnica en el campo del hogar conectado (IoT), además se ha evidenciado la falta de una arquitectura de software que incorpore la analítica de datos en estructuras no relacionales.

1.3. Justificación

La pluralidad de los datos posibilita la oportunidad para definir una arquitectura que permita almacenarlos; la manera en que es guardada la permite reducir el nivel de complejidad a la hora de analizarla y obtener información relevante. Dado esto, se hace necesario definir una arquitectura computacional de *IoT* que permita almacenar los datos de manera tal que sean procesables y relevantes para la técnica de *deep learning*; los datos no estructurados y el gran flujo de información de manera concurrente hace que además de contar con una arquitectura de almacenamiento apropiada se debe tener una infraestructura en la nube con capacidades suficientes para atender la totalidad de las peticiones, almacenar *big data* y tener suficiencia para analizarla.

Actualmente los algoritmos de aprendizaje profundo son raramente usados en hardware de clase móvil/IoT porque a menudo imponen niveles debilitantes de sobrecarga del sistema (Ejemplos: Memoria, computación y energía). Uno de los factores cruciales para poder adoptar el *deep learning* es la falta de una comprensión sistemática de cómo estos algoritmos se comportan en tiempo de inferencia en hardware de recursos limitados [Lane et al., 2015].

El mundo es dinámico y complejo; Tales condiciones a menudo confunden el procesamiento de señal y las técnicas de aprendizaje de máquina empleadas para la inferencia del sensor. El enfoque más prometedor para hacer frente a este desafío de hoy es el *deep learning*. Los avances en este campo de *machine learning* han transformado cuántas tareas de inferencia relacionadas con IoT y aplicaciones móviles se realizan [Lane et al., 2015].

En este proyecto se desea profundizar en el campo de *IoT* aplicando *deep learning*, como se ha evidenciado en diferentes fuentes bibliográficas el *deep learning* es la técnica que dadas sus características principales mejor se adapta al tipo de información a ser analizada [Lane et al., 2015, LeCun et al., 2015]. Para las empresas que fabrican electrodomésticos del *connected home* es relevante descubrir de qué manera sus clientes interactúan con sus productos, datos como la frecuencia de uso, el tiempo que tarda el cliente interactuando con los diferentes componentes, el nivel de fuerza aplicada sobre el producto entre otros, revelan a la compañía información tal como el nivel de satisfacción del cliente, el uso indebido de componentes y el tiempo de encuentro con el usuario, esta información brinda ventaja competitiva y posibles nuevas ideas de mejora y nuevos productos.

1.4. Objetivos

1.4.1. Objetivo General

Definir una arquitectura de Software que incorpore un módulo de analítica de datos, donde se implemente modelos basados en *deep learning* que sea aplicable a los datos provenientes de dispositivos *IoT* de *connected home* específicamente de *smart kitchen*.

1.4.2. Objetivos Específicos

- Establecer cuales son los algoritmos de *deep learning* que pueden ser aplicados en un contexto de IoT en el hogar conectado.

- Analizar las características de los datos de dispositivos *IoT* de hogares conectados, a los cuales se les puede aplicar la técnica de analítica de *deep*

learning.

- Implementar una arquitectura de software que permita almacenar, procesar y analizar grandes volúmenes de datos aplicando *deep learning* de dispositivos *IoT* de *connected home* específicamente de *smart kitchen*.
- Validar la arquitectura con datos que se ajusten a las características definidas.

1.5. Estructura del documento

El presente documento esta organizado de la siguiente manera: El capítulo 1 se encuentra estructurado con las secciones de introducción, campo temático, planteamiento del problema, justificación del problema y los objetivos, abordando el general y los específicos.

En el capitulo 2, se presenta la revisión bibliográfica en el marco teórico, en este se documentan referencias bibliográficas que sean relevantes para la toma de decisiones y el desarrollo del documento. Además se valida el estado del arte de todos los conceptos a ser utilizados.

En el capitulo 3, se presenta la solución propuesta de arquitectura computacional, en donde se mostrará de manera detallada el desarrollo de los conceptos previos hasta llegar a la arquitectura.
.

En el capitulo 4, se presentan los resultados de las validaciones de la arquitectura. Se documentan todas las pruebas realizadas junto a los enlaces de los repositorios de código donde se puede descargar y replicar la arquitectura propuesta.

En el capitulo 5, se entregan las conclusiones del proyecto, los resultados obtenidos y los trabajos futuros derivados de la presente solución.

Capítulo 2

Revisión Bibliográfica

Este proyecto se fundamenta en la intención concreta de analizar datos arrojados por dispositivos IoT, para ello es necesario conocer los conceptos clave que engloba el proceso y así poder tener una visión amplia del trabajo y su desarrollo.

El primer concepto a relacionar es el internet de las cosas (IoT por sus siglas en inglés), a continuación se da claridad del término:

2.1. IoT

El internet de las cosas (IoT por sus siglas en inglés) fue acuñado por primera vez por Kevin Ashton como título de una presentación en 1999 [Ding et al., 2013, Wu et al., 2014], esta se define como la infraestructura que hace los objetos remotamente accesibles y los conecta entre sí, los dispositivos conectados a IoT poseen tres características principales, la primera de ellas es que deben medir algo, deben ser autónomos y deben ser accionables para brindar control [Botta et al., 2016, Stolpe, 2016].

El Internet de las cosas se refiere a los objetos direccionables y sus representaciones virtuales en una estructura similar a Internet. Tales objetos pueden vincular información sobre ellos, o puede transmitir datos de sensores de tiempo real sobre su estado u otras propiedades útiles asociadas con objeto.

La recolección e intercambio de datos se realiza mediante software, sensores y conectividad de red, que son Incrustados en objetos físicos. La infraestructura que Hace tales objetos accesibles remotamente y los conecta, se llama Internet de las cosas (IoT). En 2010, ya 12,5 Mil Millones de dispositivos se conectaron a la IoT, un número Aproximadamente el doble de la población mundial en ese

momento (6.800 millones) [Stolpe, 2016].

El IoT consiste en objetos físicos (o çosas") que incorporan componentes electrónicos, software, sensores y componentes de comunicación, lo que les permite recopilar e intercambiar datos. Las cosas físicas ya no están separadas del mundo virtual, sino conectadas a Internet. Se puede acceder a remotamente, controlado e incluso hecho actuar.

Las ideas que se asemejan al IoT remontan al año 1988, comenzando con el campo de la computación ubicua. En 1991, Mark Weiser enmarca sus ideas para la computadora del siglo XXI. Weiser imaginaba que las computadoras eran lo suficientemente pequeñas como para desaparecer de nuestra vista, convirtiéndose en parte del fondo, para que se usen sin más reflexión. Las habitaciones recibirán más de 100 dispositivos conectados, lo que percibir su entorno, intercambiar datos y proporcionar a los seres humanos información similar a los signos físicos, notas, papel, tableros, etc. Los dispositivos necesitan auto conocimiento, por ejemplo, de su ubicación. Muchas de las ideas originales de Weiser todavía pueden encontrarse en las definiciones actuales del IoT y los requisitos para los dispositivos de acuerdo. [Stolpe, 2016].

Para resumir, Internet de las cosas es un tipo de red de algunos objetos físicos o cosas que, integrados con software, electrónica, sensores y conectividad que los habilita, logran un mayor valor y servicio al intercambiar datos con fabricantes, operadores y algunos otros dispositivos conectados.

Figura 2.1: Tecnología IoT [Stergiou et al., 2018]

¿Qué significa cuando los dispositivos y sensores están conectados en red y

se comunican entre sí? ¿Cómo puede el Internet de las cosas afectar nuestra vida diaria? Los sistemas de GPS, sistemas de alarma y termostatos, todos envían y reciben alimentaciones constantes para monitorear y automatizar las actividades en nuestra vida diaria. Y lo que no es tan obvio: mosaico, tazas, ropa y otros objetos cotidianos también pueden unirse a la red para enviar y recibir datos a través de Internet.

Las empresas examinan las oportunidades en las que la transmisión de datos creará nuevos mercados para inspirar cambios positivos o mejorar los servicios existentes. Algunos ejemplos de sectores que están en el centro de estos desarrollos se enumeran a continuación [Stergiou et al., 2018]:

1. Solución inteligente en el cubo de transporte: soluciones inteligentes en el cubo de transporte, lograr una reducción del tráfico en las carreteras, reducir el consumo de combustible, establecer prioridades en los programas de reparación de vehículos y salvar vidas.

2. Redes de energía inteligentes que incorporan más renovables: Las redes de energía inteligentes que incorporan más energías renovables mejoran la confiabilidad del sistema y reducen las cargas de los consumidores, lo que proporciona electricidad más barata.

3. Monitoreo remoto de pacientes: el monitoreo remoto de pacientes proporciona un fácil acceso a la atención médica, mejora la calidad de los servicios, aumenta el número de personas atendidas y ahorra dinero.

4. Sensores en hogares y aeropuertos: los sensores en hogares y aeropuertos, o incluso en sus zapatos o puertas, mejoran la seguridad al enviar señales cuando no se usan durante un cierto período de tiempo o cuando se usan en el momento incorrecto.

5. Sensores de monitoreo del motor que detectan y predicen problemas de mantenimiento: sensores de monitoreo del motor que detectan y predicen problemas de mantenimiento, mejoran la reposición de inventario e incluso definen prioridades en la programación de trabajos de mantenimiento, reparaciones y operaciones regionales.

6. El IoT es capaz de impulsar diversas aplicaciones médicas, como el control remoto de la salud, los programas de acondicionamiento físico, la rehabilitación, las enfermedades crónicas y el cuidado de los ancianos. [Albahri et al., 2021]

El continuo crecimiento de las aplicaciones de Internet de las cosas (IoT) está generando grandes cantidades de datos. Cada día se crean 2.5 quintillones de bytes de datos y para 2020, se estima que cada persona creará 1.7MB por segundo [Renart et al., 2019].

Al tener tantos datos en cada dispositivo, se hace necesario que estos sean transmitidos de máquina a máquina (M2M) dando lugar a protocolos de comunicación que permitan la transferencia de datos y la recepción de los mismos. Uno de los protocolos más conocido en el campo del internet de las cosas dadas sus características técnicas es MQTT que será expuesto a continuación:

2.2. MQTT

MQTT es uno de los protocolos de comunicación M2M más antiguos, que se introdujo en 1999. Fue desarrollado por Andy Stanford-Clark de IBM y Arlen Nipper de Arcom Control Systems Ltd (Eurotech). Es un protocolo de mensajería de publicación / suscripción diseñado para comunicaciones M2M ligeras en redes restringidas. El cliente MQTT publica mensajes a un agente MQTT, que están suscritos por otros clientes o pueden ser retenidos para la suscripción futura. Cada mensaje se publica en una dirección, conocida como tema. Los clientes pueden suscribirse a múltiples temas y recibir cada mensaje publicado en cada tema. MQTT es un protocolo binario y normalmente requiere un encabezado fijo de 2 bytes con pequeñas cargas de mensajes de hasta 256 MB. Utiliza TCP como protocolo de transporte y TLS / SSL por seguridad. Por lo tanto, la comunicación entre el cliente y el corredor es una conexión orientada. Otra gran característica de MQTT son sus tres niveles de calidad de servicio (QoS) para la entrega confiable de mensajes. MQTT es más adecuado para grandes redes de dispositivos pequeños que necesitan ser monitoreados o controlados desde un servidor de fondo en Internet. No está diseñado para la transferencia de dispositivo a dispositivo ni para datos de multidifusión a muchos receptores. Es un protocolo de mensajería muy básico que ofrece sólo unas pocas opciones de control [Naik, 2017].

MQTT, en virtud de su arquitectura inherente de publicación-suscripción, es muy útil para las aplicaciones tipo ”PUSH”para liberar a los dispositivos restringidos de las operaciones que necesitan recursos como ”sondeo”para obtener los datos actualizados. Sin embargo, en el caso de puertas de enlace de borde restringidas, este protocolo se puede utilizar para ”publicar”datos de sensores agregados en tiempo real al servidor web de forma asincrónica y confiable [Bandyopadhyay and Bhattacharyya, 2013].

Uno de los protocolos de comunicación con el que comúnmente es comparado MQTT es CoAP. CoAP es un protocolo de transferencia web basado en la arquitectura REST (Representational State Transfer). Es muy similar a HTTP. Sin embargo, para evitar la sobrecarga en TCP como protocolos orientados a la conexión, CoAP está originalmente diseñado para usarse sobre UDP con disposiciones para un conjunto reducido de mecanismos de confiabilidad como TCP. La confiabilidad opcional es compatible con una capa de mensajería lógica en la parte superior de UDP. Lógicamente, CoAP puede considerarse como un traje de protocolo único de dos capas lógicas: capa de solicitud / respuesta en la parte

superior y capa de mensajería en la parte inferior para interactuar con el UDP. Admite 4 categorías de mensajes: confirmable, no confirmable, mensaje de reconocimiento y restablecimiento. Los mensajes de tipo confirmables se utilizan para una comunicación confiable. El mensaje confirmable va acompañado de un acuse de recibo del nodo receptor para proporcionar confiabilidad.

En la tabla 2.1 se encuentra la comparativa entre los dos protocolos mostrando las características de cada uno de ellos.

Con la relevancia que el internet de las cosas está tomando y la gran cantidad de datos que está generando, se debe explotar la totalidad de su capacidad, esto cobra sentido para el mundo real si sus datos son transmitidos, analizados y procesados, para aprender del entorno y lograr realizar una comunicación fluida y útil de máquina a máquina (M2M), para lograr este objetivo se utilizan técnicas tales como el machine learning que será expuesto a continuación:

2.3. Machine Learning

EL aprendizaje de máquina (ML) se introdujo a finales de 1950 como una técnica de inteligencia artificial (AI). Con el tiempo, su enfoque evolucionó y se desplazó más hacia algoritmos que son computacionalmente viables y robustos. En la última década, las técnicas de aprendizaje automático se han utilizado extensamente para una amplia gama de tareas incluyendo clasificación, regresión y estimación de densidad en una variedad de áreas de aplicación como bioinformática, reconocimiento de voz, detección de spam, visión por computadora, detección de fraudes y redes publicitarias. Los algoritmos y técnicas utilizados provienen de muchos campos diversos, incluyendo estadísticas, matemáticas, neurociencia y ciencias de la computación [Alsheikh et al., 2014].

La tecnología de aprendizaje de máquina potencia muchos aspectos sociedad: desde las búsquedas en la web hasta el filtrado de contenidos en las redes sociales recomendaciones sobre sitios web de comercio electrónico, y está cada vez más presente en productos de consumo tales como cámaras y smartphones Los sistemas de aprendizaje automático se utilizan para identificar objetos en imágenes, transcribir el habla en texto, hacer coincidir noticias, publicaciones o productos con los intereses de los usuarios, y seleccione los resultados relevantes de la búsqueda.

Al aprovechar herramientas matemáticas y estadísticas complejas, ML hace que las máquinas sean capaces de realizar tareas intelectuales independientes que los seres humanos han resuelto tradicionalmente [Musumeci et al., 2018]. Esta idea de automatizar tareas complejas ha generado un gran interés en campos como el de las redes de comunicación computacionales, con la expectativa de que varias actividades involucradas en el diseño y operación de redes de co-

municación puedan descargarse a las máquinas. Algunas aplicaciones de ML en diferentes áreas de redes ya han cumplido con estas expectativas en áreas como la detección de intrusos [Buczak and Guven, 2015], clasificación de tráfico [Nguyen and Armitage, 2008], las radios cognitivas [Bkassiny et al., 2012] entre otros.

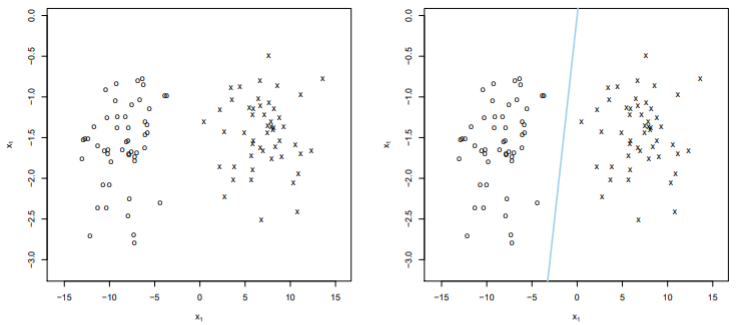
Machine learning cuenta con diferentes tipos de algoritmos para su implementación que han sido creados y mejorados desde el inicio del uso de esta técnica que pretende dar provecho a todos los recursos computacionales [Jaramillo Garzón, 2013]. A continuación se presentan las categorías de machine learning comúnmente conocidas [Marsland, 2015]:

- **Aprendizaje supervisado (Supervised Learning):** se ofrece un conjunto de ejemplos de capacitación con objetivos adecuados y, sobre la base de este conjunto de capacitación, los algoritmos responden correctamente a todas las entradas factibles. Aprender de los ejemplos es otro nombre de Aprendizaje supervisado. La clasificación y la regresión son los tipos de aprendizaje supervisado [Marsland, 2015]. Los algoritmos de aprendizaje supervisado tienen acceso a un conjunto de entrenamiento etiquetado que consiste en pares (característica, etiqueta), denotados por $\{x_i, y_i\}_{i=1}^{N}$. En esta configuración, los vectores de características están nuevamente representados por los vectores D-dimensionales $x_i \in \mathbb{R}^D$, mientras que las etiquetas son las predicciones deseadas para cada instancia. Cuando las etiquetas son variables continuas, es decir, $y \in \mathbb{R}$, el modelo construido es un regresor. A su vez, cuando las predicciones se limitan a un conjunto finito de etiquetas discretas, $Y \in Y_{j\,j=1}^{C}$, el modelo entrenado es un clasificador. Entonces, dicho clasificador es una función matemática $f(x)$, que asocia cada vector de características con su correspondiente etiqueta verdadera [Jaramillo Garzón, 2013]:

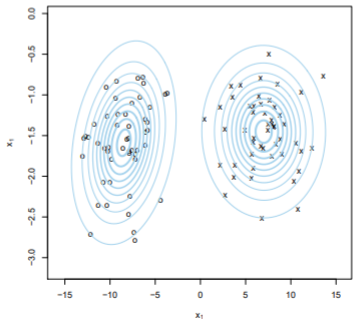
  $$f : X \rightarrow y$$

  La Figura 2.2 muestra un ejemplo de un problema de clasificación supervisada de dos clases. En este contexto, se pueden utilizar dos enfoques para derivar la función de decisión $f(x)$: discriminativa y generativa. Los clasificadores discriminativos se centran en calcular la frontera de decisión entre las clases como en la Figura 2.2 (b), donde se usa un clasificador discriminante lineal. Los clasificadores generativos, por otro lado, se centran en modelar los datos para obtener un modelo por clase y proporcionar probabilidades de membresía para nuevas instancias. La figura 2.2 (c) representa los niveles de contorno de dos distribuciones de probabilidad gaussianas ajustadas a los datos (Jaramillo Garzón, J. A., 2013).

  Clasificación: da la predicción de Sí o No, por ejemplo, “¿Es este tumor



(a) Datos (b) Clasificador Discriminativo



(c) Clasificador Generativo

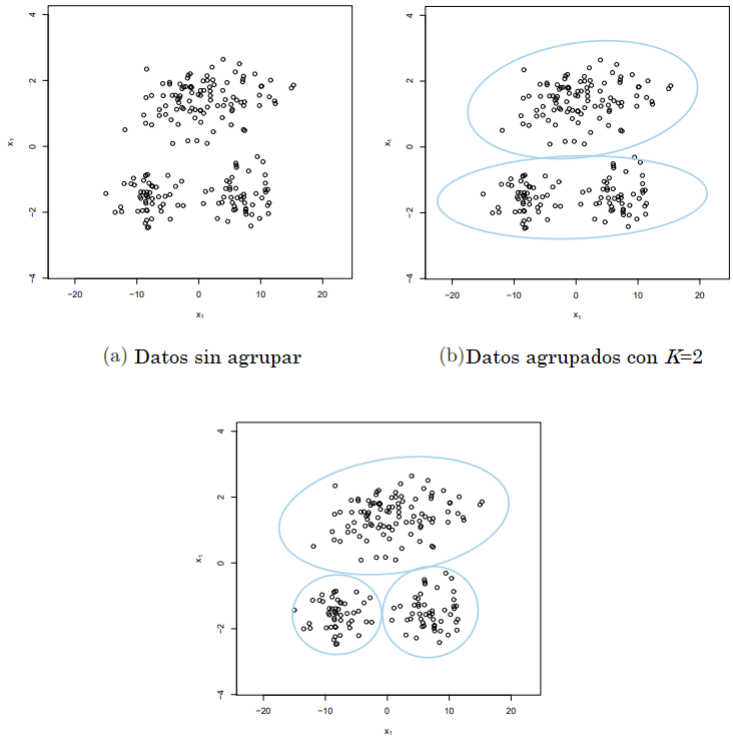
Figura 2.2: Ejemplo de aprendizaje supervisado [Jaramillo Garzón, 2013]

canceroso?", "¿Esta cookie cumple con nuestros estándares de calidad?".

Regresión: da la respuesta de "cuánto" y "cuántos".

- **Aprendizaje no supervisado (Unsupervised Learning):** no se proporcionan respuestas u objetivos correctos. La técnica de aprendizaje no supervisado intenta descubrir las similitudes entre los datos de entrada y, basándose en estas similitudes, la técnica de aprendizaje no supervisado clasifica los datos. Esto también se conoce como estimación de densidad. El aprendizaje no supervisado contiene agrupaciones [Marsland, 2015]. En el aprendizaje no supervisado, el sistema observa un conjunto no etiquetado de elementos representados por sus vectores de características D-dimensionales $x_i{}_{i=1}^{N}, x_i \in R^D$, extraídos de un espacio de características X. El objetivo principal es descubrir su distribución subyacente en orden para agruparlos en k grupos. En este contexto, no hay una respuesta "correcta", ya que no existe un conocimiento previo sobre la mem-

bresía correcta de las muestras (por lo que este paradigma se denomina no supervisado)[Jaramillo Garzón, 2013].



(a) Datos sin agrupar (b)Datos agrupados con K=2

(c) Datos agrupados con K=3

Figura 2.3: Ejemplo de aprendizaje no supervisado [Jaramillo Garzón, 2013]

La Figura 2.3 muestra un ejemplo del resultado de un alumno sin supervisión para dos (Figura 2.3 (b)) y tres grupos (Figura 2.3 (c)). Como no hay supervisión, ambos esquemas de agrupamiento podrían ser soluciones válidas para diferentes tareas [Jaramillo Garzón, 2013].

Agrupación: crea agrupaciones en función de la similitud.

Aprendizaje semi supervisado (Semi-supervised Learning): la técnica de aprendizaje semi supervisado es una clase de técnicas de aprendizaje supervisado. Este aprendizaje también utilizó datos no etiquetados para fines de capacitación (generalmente una cantidad mínima de datos etiquetados con una gran cantidad de datos no etiquetados). El aprendizaje semi-supervisado se encuentra entre el aprendizaje no supervisa-

do (datos no etiquetados) y el aprendizaje supervisado (datos etiquetados) [Marsland, 2015]. Como su nombre lo indica, el aprendizaje semi-supervisado se encuentra entre el aprendizaje no supervisado y supervisado. De hecho, la mayoría de las estrategias de aprendizaje semi-supervisado se basan en extender el aprendizaje supervisado o no supervisado para incluir información adicional típica del otro paradigma de aprendizaje. Específicamente, el aprendizaje semi-supervisado abarca varios entornos diferentes, que incluyen [Zhu and Goldberg, 2009]:

- **Clasificación semi-supervisada:** También conocida como clasificación con datos etiquetados y no etiquetados (o datos parcialmente etiquetados), esta es una extensión del problema de clasificación supervisada. Los datos de entrenamiento consisten en l instancias etiquetadas $\{(x_i, y_i)\}_{i=1}^{l}$ y u instancias no etiquetadas $\{(x_j)\}_{j=l+1}^{l+u}$. Normalmente, se supone que hay muchos más datos sin etiquetar que datos etiquetados, es decir, $u \gg l$. El objetivo de la clasificación semi-supervisada es entrenar a un clasificador f a partir de los datos etiquetados y no etiquetados, de modo que sea mejor que el clasificador supervisado entrenado solo en los datos etiquetados [Zhu and Goldberg, 2009].
- **Agrupamiento restringido:** Esta es una extensión para la agrupación no supervisada. Los datos de entrenamiento consisten en instancias sin etiquetar $\{x_i\}_{j=1}^{n}$, así como también ïnformación supervisada"sobre los grupos. Por ejemplo, dicha información puede denominarse restricciones de enlace obligatorio, que dos instancias x_i, x_j deben estar en el mismo clúster; y no se pueden vincular las restricciones, que x_i, x_j no pueden estar en el mismo clúster. También se puede restringir el tamaño de los grupos. El objetivo de la agrupación restringida es obtener una mejor agrupación que la agrupación de datos no etiquetados solos [Zhu and Goldberg, 2009].

- **Aprendizaje de refuerzo (Reinforcement Learning):** este aprendizaje es fomentado por la psicología conductista. El algoritmo se informa cuando la respuesta es incorrecta, pero no informa cómo corregirla. Tiene que explorar y probar varias posibilidades hasta que encuentre la respuesta correcta. También se conoce como aprender con crítica. No recomienda mejoras. El aprendizaje de refuerzo es diferente del aprendizaje supervisado en el sentido de que no se ofrecen conjuntos de entrada y salida precisos, ni acciones subóptimas claramente precisas. Además, se centra en el rendimiento en línea [Marsland, 2015]. El aprendizaje de refuerzo (RL) se refiere tanto a un problema de aprendizaje como a un subcampo de aprendizaje automático. Como un problema de aprendizaje, se refiere a aprender a controlar un sistema para maximizar algún valor numérico que represente un objetivo a largo plazo. A continuación se presenta un ejemplo en el que se tiene un entorno típico donde opera el aprendizaje por refuerzo: un controlador recibe el estado del sistema controlado y

una recompensa asociada con la última transición de estado. Luego calcula una acción que se envía de vuelta al sistema. En respuesta, el sistema hace una transición a un nuevo estado y el ciclo se repite. El problema es aprender una forma de controlar el sistema para maximizar la recompensa total. Los problemas de aprendizaje difieren en los detalles de cómo se recopilan los datos y cómo se mide el rendimiento [Szepesvári, 2010].

- **Aprendizaje evolutivo (Evolutionary Learning):** este aprendizaje de evolución biológica se puede considerar como un proceso de aprendizaje: los organismos biológicos se adaptan para avanzar en sus tasas de supervivencia y la posibilidad de tener manantiales. Al usar la idea de aptitud física, para verificar cuán precisa es la solución, podemos usar este modelo en una computadora.
- **Aprendizaje profundo (Deep Learning):** esta rama del aprendizaje automático se basa en un conjunto de algoritmos. En datos, estos algoritmos de aprendizaje modelan la abstracción de alto nivel. Utiliza gráficos profundos con varias capas de procesamiento, formadas por muchas transformaciones lineales y no lineales [Fatima et al., 2017]. Esta categoría se ampliará en detalle en adelante en el documento,

Luego de repasar algunas categorías de machine learning, es necesario ver una serie de técnicas que permiten su aplicación, a continuación se describirán algunas de ellas hasta llegar a aquella que, según la revisión bibliográfica y las pruebas realizadas, mejor se ajusta a los requerimientos específicos del caso de estudio probado en este documento.

2.3.1. Máquinas de soporte vectorial

Las Máquinas de Soporte Vectorial (SVM) es otra de las técnicas utilizadas, la cual aprende a predecir una nueva clase de muestra a partir de unos ejemplos, esta técnica está basada en la minimización de riesgo estructural SRM (Structural Risk Minimization) [Betancourt, 2005] la cual se ha utilizado para la clasificación automática de señales sísmicas [Hurtado et al., 2002], para identificar fracturas naturales en pozos de yacimientos de hidrocarburos el cual mostró exactitud entre 72,3 % y 82,2 % en 5 pozos evaluados del campo estudiado [Leal et al., 2016], por otro lado se ha usado para el reconocimiento de patrones utilizando huellas dactilares de descargas parciales usando el filtrado de wavelet los cuales mostraron altas tasas de reconocimiento [Guzmán et al., 2017], Así mismo se han implementado SVM junto con los métodos de Kernel para mejorar la inferencia transductiva de motores de búsqueda de texto [Espinosa-Oviedo et al., 2017], y por último se ha implementado SVM para el reconocimiento de voz para emular la toma de decisiones de los humanos (tarea de clasificación y análisis de audio realizada por los técnicos), el cual mostró un porcentaje de clasificación exitosa en comparación con una plataforma automática llamada CheckMyRou-

tes [Wilches-Cortina et al., 2017].

Las SVM son ampliamente utilizadas en la clasificación binaria ya que su idea original es encontrar un hiperplano para separar las dos clases con el máximo margen. la principal ventaja de las SVM es su habilidad de clasificación para resolver los problemas no lineales usando la función kernel. [Di et al., 2019]. Aunque originariamente las SVMs fueron pensadas para resolver problemas de clasificación dirigida binaria, su aplicación se ha extendido a problemas de clasificación múltiple y regresión [Castañeda et al., 2019].

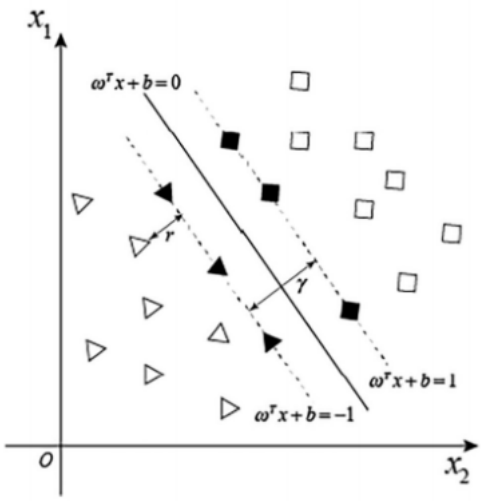


Figura 2.4: Ilustración de SVM en un conjunto de datos de dos características. Los puntos llenos en las líneas punteadas indican los vectores de soporte [Di et al., 2019]

Las Máquinas de Vector Soporte se fundamentan en el Maximal Margin Classifier, que, a su vez, se basa en el concepto de hiperplano [Castañeda et al., 2019]. Uno de los mejores usos que tienen las máquinas de soporte vectorial es la clasificación que ha demostrado ser al menos tan preciso como otros clasificadores ampliamente utilizados en sensores remotos [Foody and Mathur, 2006]. Con SVM, los casos de entrenamiento más útiles son aquellos que se encuentran cerca del lugar donde se colocará el hiperplano [Foody, 2004, Pal and Mather, 2005]. Por lo tanto, para la clasificación por técnicas como SVM, se pueda utilizar una pequeña muestra de entrenamiento, basada en píxeles mixtos, para obtener una clasificación precisa.[Foody and Mathur, 2006].

Los métodos tradicionales de aprendizaje automático, por ejemplo *Naive Bayes*, *Random Forests* y *Decision Tree*, se han aplicado ampliamente en la clasificación de la identificación del tráfico. *Restricted Boltzmann machine* (RBM) mez-

clado con SVM para detectar e identificar el tráfico de la red. [Zhong et al., 2021]

Otra de las técnicas utilizadas dentro del campo del machine learning es el clustering haciendo uso de kmeans que será descrita a continuación.

2.3.2. Clustering con K-Means

La agrupación de K-means es un método comúnmente utilizado para particionar automáticamente un conjunto de datos en k grupos. Continúa seleccionando k centros de clúster iniciales y luego refinándolos iterativamente de la siguiente manera:

- Cada instancia d_i se asigna a su centro de cluster más cercano
- Cada centro de clúster C_j se actualiza para ser la media de sus instancias constituyentes.

El algoritmo converge cuando no hay más cambios en la asignación de instancias a clústeres. En este trabajo, inicializamos los clústeres utilizando instancias elegidas al azar del conjunto de datos. Los conjuntos de datos que utilizamos se componen únicamente de características numéricas o simbólicas. Para las características numéricas, utilizamos una métrica de distancia euclidiana; para características simbólicas, calculamos la distancia de Hamming.

El último problema es cómo elegir k. Para los conjuntos de datos donde ya se conoce el valor óptimo de k (es decir, todos los conjuntos de datos UCI), lo usamos; Para el problema del mundo real de encontrar carriles en los datos del GPS, utilizamos una búsqueda envolvente para localizar el mejor valor de k [Wagstaff et al., 2001].

Para un conjunto de puntos $X = \{x_1; ...; x_n\}; x_i \in R^d$, el algoritmo de K-Means crea K-particiones $\{X_l\}_{l=1}^{K}$ de k entonces si $\{\mu_1, ..., \mu_k\}$representa los centros de partición k, Entonces la siguiente es la función objetivo:

$$J_{kmeans} = \sum_{l=1}^{k} \sum_{x_i \in X_l} \|X_i - l\|^2$$

es localmente minimizada [Basu et al., 2002].

En su forma básica, el problema de agrupamiento se define como el problema de agrupar grupos homogéneos de puntos de datos en un conjunto de datos dado. Cada uno de estos grupos se denomina clúster y puede definirse como

una región en la que la densidad de los objetos es localmente más alta que en otras regiones [Likas et al., 2003].

Algunas aplicaciones de clustering con K-Means se pueden apreciar a continuación:

- **Particionar el dataset:** K-means es uno de los algoritmos de aprendizaje no supervisados más simples que resuelven el conocido problema de agrupamiento. El procedimiento sigue una manera simple y fácil de clasificar un conjunto de datos dado a través de un cierto número de grupos (supongamos que k grupos) fijados a priori. La idea principal es definir k centroides, uno para cada grupo. Estos centroides deben colocarse de una manera astuta porque la ubicación diferente causa resultados diferentes. Entonces, la mejor opción es colocarlos lo más lejos posible el uno del otro. El siguiente paso es tomar cada punto que pertenece a un conjunto de datos determinado y asociarlo al centroide más cercano. Cuando no hay ningún punto pendiente, se completa el primer paso y se realiza una edad temprana del grupo. En este punto, tenemos que volver a calcular k nuevos centroides como baricentros de los grupos resultantes del paso anterior. Después de tener estos k nuevos centroides, se debe hacer un nuevo enlace entre los mismos puntos de ajuste de datos y el nuevo centroide más cercano. Se ha generado un bucle. Como resultado de este bucle, podemos notar que los k centroides cambian su ubicación paso a paso hasta que no se realicen más cambios. En otras palabras, los centroides ya no se mueven. La agrupación de K-medias genera un número específico de grupos disjuntos, planos (no jerárquicos). Es muy adecuado para generar clusters globales. El método K-Means es numérico, no supervisado, no determinista e iterativo. [Jipkate and Gohokar, 2012].

Teniendo clara la intención global del machine learning y ver algunas de sus técnicas, se procede a elegir una de ellas, siendo esta la que mejor se adaptan a IoT [Lane et al., 2015, LeCun et al., 2015], de acuerdo a esto se procede a definir la técnica de deep learning:

2.3.3. Deep Learning

El aprendizaje es un fenómeno polifacético. Los procesos de aprendizaje incluyen la adquisición de nuevos conocimientos declarativos, el desarrollo de habilidades motoras y cognitivas a través de la instrucción o la práctica, la organización de nuevos conocimientos en representaciones generales y efectivas y el descubrimiento de nuevos hechos y teorías a través de la observación y la experimentación. Desde el inicio de la era de la computadora, los investigadores se han esforzado por implementar tales capacidades en las computadoras. Resolver este problema ha sido, y sigue siendo, el objetivo más desafiante y fascinante de largo alcance en la inteligencia artificial (AI). El estudio y el modelado computacional de los procesos de aprendizaje en sus múltiples manifestaciones es el

tema de importancia del machine learning. [Carbonell et al., 1983].

Machine learning potencia muchos aspectos de la sociedad moderna, desde las búsquedas en la web hasta el filtrado de contenidos en las redes sociales y las recomendaciones sobre sitios web de comercio electrónico, y está cada vez más presente en productos de consumo como cámaras y teléfonos inteligentes. Los sistemas de aprendizaje automático se utilizan para identificar objetos en imágenes, transcribir el habla en texto, combinar noticias, publicaciones o productos con los intereses de los usuarios y seleccionar resultados relevantes de búsqueda. Cada vez más, estas aplicaciones hacen uso de una clase de técnica llamada aprendizaje profundo (Deep Learning)[LeCun et al., 2015]. En la figura 2.5 se representa una red neuronal multicapa que hace uso del algoritmo de backpropagation.

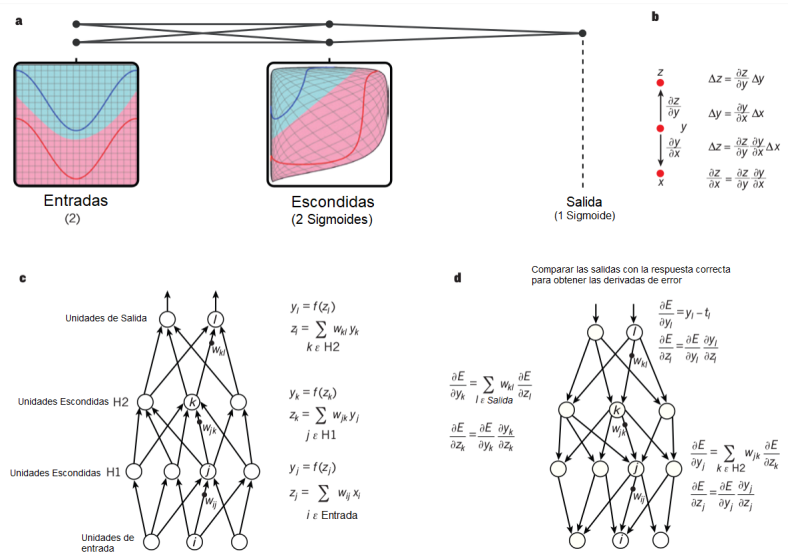


Figura 2.5: Red neuronal multicapa y backpropagation [LeCun et al., 2015]

1. Una red neuronal multicapa (mostrada por los puntos conectados) puede distorsionar el espacio de entrada para hacer que las clases de datos (ejemplos de las cuales están en las líneas roja y azul) se pueden separar linealmente. Observe cómo una cuadrícula regular (mostrada a la izquierda) en el espacio de entrada también se transforma (mostrada en el panel central) por unidades ocultas. Este es un ejemplo ilustrativo con sólo dos unidades de entrada, dos unidades ocultas y una unidad de salida, pero las redes utilizadas para el reconocimiento de objetos o el procesamiento de lenguaje natural contienen decenas o cientos de miles de unidades.

Reproducido con permiso de C. Olah (http://colah.github.io/).

2. La regla de la cadena de los derivados nos dice cómo se componen dos efectos pequeños (el de un pequeño cambio de x en y, y el de y en z). Un pequeño cambio $\triangle x$ en x se transforma primero en un pequeño cambio $\triangle y$ en y al multiplicarse por $\frac{\partial y}{\partial x}$ (es decir, la definición de derivada parcial). Similarmente, el cambio $\triangle y$ crea un cambio $\triangle z$ en z. La sustitución de una ecuación por la otra da la regla de la cadena de los derivados - cómo $\triangle x$ se convierte en $\triangle z$ mediante la multiplicación por el producto de $\frac{\partial y}{\partial x}$ y $\frac{\partial z}{\partial x}$. También funciona cuando x, y y z son vectores (y las derivadas son matrices Jacobianas).

3. Las ecuaciones utilizadas para calcular el paso directo en una red neuronal con dos capas ocultas y una capa de salida, cada una constituyendo un módulo a través del cual se pueden retro propagar gradientes. En cada capa, primero calculamos la entrada total z a cada unidad, que es una suma ponderada de las salidas de las unidades en la capa de abajo. Entonces se aplica una función no lineal $f(.)$ a z para obtener la salida de la unidad. Para simplificar, hemos omitido los términos de sesgo.

   Las funciones no lineales utilizadas en las redes neuronales incluyen la unidad rectificada lineal (ReLU) $f(Z) = max(0, z)$, comúnmente utilizada en los últimos años, así como los sigmoideos más convencionales, como la tangente hiperbólica $f(Z) = \frac{exp(z)-exp(-z)}{exp(z)+exp(-z)}$ y la función logística, $f(Z) = \frac{1}{1+exp(-z)}$.

4. Las ecuaciones utilizadas para calcular el paso hacia atrás. En cada capa oculta calculamos la derivada de error con respecto a la salida de cada unidad, que es una suma ponderada de las derivadas de error con respecto a las entradas totales a las unidades en la capa anterior. A continuación convertimos la derivada de error con respecto a la salida en la derivada de error con respecto a la entrada multiplicándola por el gradiente de$f(Z)$.

   En la capa de salida, la derivada de error con respecto a la salida de una unidad se calcula diferenciando la función de coste. Esto da $yl - tl$ si la función de coste para la unidad l es 0.5 $(yl - tl)$ 2, donde tl es el valor objetivo. Una vez que se conoce la $\frac{\partial E}{\partial z}k$, la derivada de error para el peso W_{jk} en la conexión de la unidad j en la capa inferior es simplemente $yj\frac{\partial E}{\partial z}k$.

A pesar de los grandes avances que el aprendizaje de máquina ha tenido durante un largo periodo de tiempo, sufría de algunas limitaciones importantes con respecto a los datos de entrada y la experiencia requerida por los usuarios

que hacían uso de estas herramientas, los datos de entrada debían ser procesados a estructuras que la máquina pudiese procesar.

El aprendizaje de representación es un conjunto de métodos que permiten alimentar a una máquina con datos sin procesar y descubrir automáticamente las representaciones necesarias para la detección o clasificación. Los métodos de aprendizaje profundo son métodos de representación-aprendizaje con múltiples niveles de representación, obtenidos mediante la composición de módulos simples pero no lineales que transforman la representación en un nivel (comenzando con la entrada en bruto) en una representación a un nivel más alto y ligeramente más abstracto . Con la composición de tales transformaciones, se pueden aprender funciones muy complejas. Para tareas de clasificación, las capas superiores de representación amplifican aspectos de la entrada que son importantes para la discriminación y suprimen variaciones irrelevantes.

El aprendizaje profundo (Deep learning como es conocido comúnmente por su traducción al inglés) permite que los modelos computacionales que están compuestos de múltiples capas de procesamiento aprendan representaciones de datos con múltiples niveles de abstracción. Estos métodos han mejorado dramáticamente el estado de la técnica en reconocimiento de voz, reconocimiento de objetos visuales, detección de objetos y muchos otros dominios como el descubrimiento de fármacos y la genómica. El aprendizaje profundo descubre la estructura intrincada en grandes conjuntos de datos utilizando el algoritmo backpropagation para indicar cómo una máquina debe cambiar sus parámetros internos que se utilizan para calcular la representación en cada capa de la representación en la capa anterior. Las redes convolucionales profundas han traído consigo avances en el procesamiento de imágenes, video, mientras que las redes recurrentes han iluminado los datos secuenciales como el texto y el habla [LeCun et al., 2015].

Las redes neuronales profundas explotan la propiedad de que muchas señales naturales son jerarquías de composición, en las que las características de nivel superior se obtienen al componer las de nivel inferior. En las imágenes, las combinaciones locales de bordes forman motivos, los motivos se agrupan en partes y las partes forman objetos. Existen jerarquías similares en el habla y el texto desde sonidos a teléfonos, fonemas, sílabas, palabras y oraciones. La agrupación permite que las representaciones varíen muy poco cuando los elementos de la capa anterior varían en posición y aspecto. Como se puede observar en la figura 2.6, en el aprendizaje profundo la clave de su buen funcionamiento son las capas ocultas, en ellas es donde la información es transmitida, analizada y finalmente clasificada [LeCun et al., 2015].

Deep Learning es un tipo de machine learning que hace que las computadoras aprendan de la experiencia y el conocimiento sin programación explícita y extraigan patrones útiles de datos sin procesar. Para los algoritmos machine learning convencionales, es difícil extraer características bien representadas

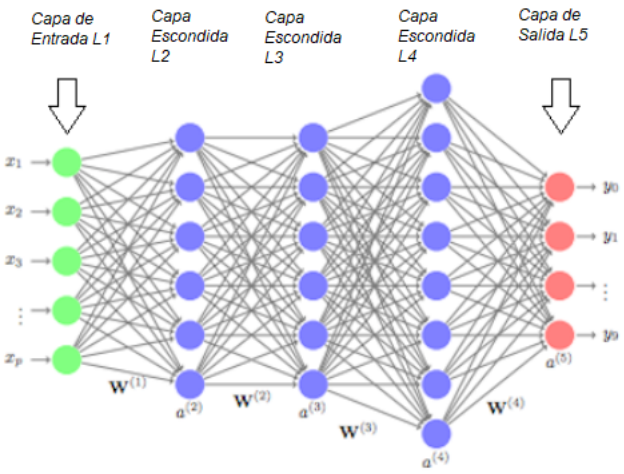


Figura 2.6: Red Neuronal con Deep Learning [LeCun et al., 2015]

debido a limitaciones, como la maldición de la dimensionalidad, el cuello de botella computacional y el requisito del dominio y el conocimiento experto. Deep Learning resuelve el problema de la representación construyendo múltiples características simples para representar un concepto sofisticado.

Por ejemplo, un sistema de clasificación de imágenes basado en Deep Learning representa un objeto al describir bordes, telas y estructuras en las capas ocultas. Con el creciente número de datos de entrenamiento disponibles, Deep Learning se vuelve más poderoso. Los modelos Deep Learning han resuelto muchos problemas complicados con la ayuda de la aceleración de hardware en tiempo computacional.

Una capa de red neuronal está compuesta por un conjunto de perceptrones (neuronas artificiales). Cada perceptrón asigna un conjunto de entradas a valores de salida con una función de activación. La función de una red neuronal se forma en una cadena

$$f(x) = f(k)(...f(2)(f(1)(x)))$$

donde $f(i)$ es la función de la capa i-ésima de la red,

$$i = 1, 2, ..., k$$

Las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN) son las dos redes neuronales más utilizadas en las arquitecturas de redes neuronales recientes. Las CNN implementan operaciones de convolución en capas ocultas para compartir pesos y reducir el número de parámetros. Las CNN pueden extraer información local de datos de entrada de tipo cuadrícula. Las CNN han demostrado éxitos increíbles en tareas de visión por computadora, como la clasificación de imágenes, detección de objetos , reconocimiento de texto y segmentación semántica.

Los RNN son redes neuronales para procesar datos de entrada secuenciales con longitud variable. Los RNN producen salidas en cada paso de tiempo. La neurona oculta en cada paso de tiempo se calcula en función de los datos de entrada actuales y las neuronas ocultas en el paso de tiempo anterior. La memoria a largo plazo a largo plazo y la unidad recurrente cerrada con puertas controlables están diseñadas para evitar la desaparición / explosión de gradientes de RNN en dependencia a largo plazo [Yuan et al., 2019].

El aprendizaje profundo puede ser usado a través de diferentes tipos de redes neuronales, a continuación se dará un resumen acerca de dos de ellas.

2.3.4. Redes neuronales convolucionales (ConvNets)

Existe un tipo particular de red profunda, que es mucho más fácil de entrenar y generaliza mucho mejor que las redes con conectividad completa entre capas adyacentes. Esta es la red neuronal convolucional. Este tipo de red neuronal logró muchos éxitos prácticos durante el período en que las redes neuronales no estaban a favor y recientemente ha sido ampliamente adoptado por la comunidad de la informática [LeCun et al., 2015].

Las redes convolucionales están diseñadas para procesar datos que vienen en forma de múltiples matrices, por ejemplo una imagen en color compuesta por tres matrices 2D que contienen intensidades de píxeles en los tres canales de color. Muchas modalidades de datos están en forma de múltiples matrices: 1D para señales y secuencias, incluyendo el lenguaje; 2D para imágenes o espectrogramas de audio; y 3D para vídeo o imágenes volumétricas. Existen cuatro ideas clave detrás de ConvNets que aprovechan las propiedades de las señales naturales: conexiones locales, pesos compartidos, agrupación y el uso de muchas capas.

La arquitectura de un ConvNet típico se estructura como una serie de etapas. Las primeras etapas se componen de dos tipos de capas: capas convolucionales y capas de agrupación. Las unidades en una capa convolucional se organizan en mapas de características, dentro de los cuales cada unidad está conectada a

parches locales en los mapas de características de la capa anterior a través de un conjunto de pesos denominado banco de filtros. El resultado de esta suma ponderada local se pasa entonces a través de una no linealidad tal como una ReLU. Todas las unidades de un mapa de funciones comparten el mismo banco de filtros. Diferentes mapas de características en una capa utilizan bancos de filtros diferentes. La razón de esta arquitectura es doble. En primer lugar, en los datos de matriz, como las imágenes, los grupos locales de valores suelen estar altamente correlacionados, formando motivos locales distintivos que se detectan fácilmente. En segundo lugar, las estadísticas locales de imágenes y otras señales son invariantes a la ubicación. En otras palabras, si un motivo puede aparecer en una parte de la imagen, podría aparecer en cualquier lugar, de ahí la idea de unidades en diferentes lugares compartiendo los mismos pesos y detectando el mismo patrón en diferentes partes de la matriz. Matemáticamente, la operación de filtrado realizada por un mapa de características es una convolución discreta, de ahí el nombre [LeCun et al., 2015].

Su capacidad se puede controlar variando su profundidad y amplitud, y también hacen suposiciones fuertes y en su mayoría correctas sobre la naturaleza de las imágenes (es decir, stationarity de la estadística y localidad de dependencias del pixel). Por lo tanto, en comparación con las redes neuronales de feedforward estándar con capas de tamaño similar, CNNs tienen mucho menos conexiones y parámetros y por lo tanto son más fáciles de entrenar, mientras que su rendimiento teóricamente mejor es sólo ligeramente peor. A pesar de las cualidades atractivas de las CNN, ya pesar de la relativa eficiencia de su arquitectura local, todavía resultaban prohibitivamente caras aplicarlas a gran escala a imágenes de alta resolución. Por suerte, las GPU actuales, emparejadas con una implementación altamente optimizada de la convolución 2D, son lo suficientemente potentes como para facilitar el entrenamiento de CNNs interesantes y los conjuntos de datos recientes como ImageNet contienen suficientes ejemplos etiquetados para entrenar a tales modelos sin sobregrabar [Krizhevsky et al., 2012].

En la figura 2.7, se detalla la arquitectura de una CNN, mostrando explícitamente la delimitación de responsabilidades entre las dos GPU. Una GPU ejecuta las partes de capa en la parte superior de la figura mientras que la otra ejecuta las partes de capa en la parte inferior. Las GPU se comunican sólo en ciertas capas [Krizhevsky et al., 2012].

Como conclusión a la información anteriormente ilustrada, se define que las redes convolucionales son altamente usadas en el reconocimiento de imágenes y como característica principal de los datos de entrada es que deben ser múltiples matrices, ejemplos de datos que cumplen con dicha característica son: 1D señales y secuencias, incluyendo el lenguaje; 2D imágenes o espectrogramas de audio; y 3D video o imágenes volumétricas.

Las CNN que se inspiran en el sistema visual humano son similares a las redes neuronales clásicas. Esta arquitectura se ha diseñado específicamente sobre

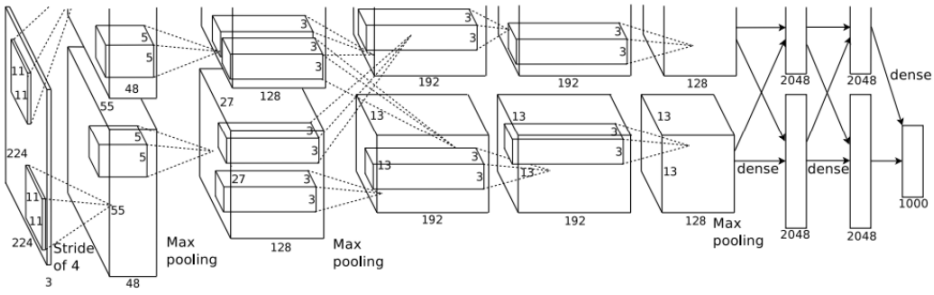


Figura 2.7: Una ilustración de una arquitectura CNN, que muestra explícitamente la delineación de responsabilidades entre las dos GPU. Una GPU ejecuta las partes de la capa en la parte superior de la figura, mientras que la otra ejecuta las partes de la capa en la parte inferior. Las GPU se comunican sólo en ciertas capas. La entrada de la red es 150,528-dimensional, y el número de neuronas en las capas restantes de la red está dado por 290, 400-186, 624-64, 896-64, 896-43, 264-4096-4096-1000 [Krizhevsky et al., 2012]

la base de la suposición explícita de que los datos brutos se componen de imágenes bidimensionales que permiten codificar ciertas propiedades al tiempo que reducen la cantidad de hiper parámetros. La topología de las CNN utiliza relaciones espaciales para reducir el número de parámetros que se deben aprender, mejorando así el entrenamiento general de retro expansión de feed-forward. La ecuación 2.1 demuestra cómo el componente de gradiente para un peso dado se calcula en el paso de retropropagación, donde E es la función de error, y es la neurona $N_{i,j}$, x es la entrada, l número de capas representativas, w es el peso del filtro con índices a y b, N es el número de neuronas en una capa dada, y m es el tamaño del filtro.

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{l}} \frac{\partial x_{ij}^{l}}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{l}} y_{(i+a)(j+b)}^{l-1} \tag{2.1}$$

Ecuación 2.1 En capas convolucionales, el componente de gradiente de un peso dado se calcula aplicando la regla de la cadena. Las derivadas parciales del error para la función de costo con respecto al peso se calculan y se usan para actualizar el peso.

La ecuación 2.2 describe el error de propagación inversa para la capa anterior utilizando la regla de la cadena. Esta ecuación es similar a la definición de convolución, donde $x_{(i+a)(j+b)}$ se reemplaza por $x_{(i-a)(j-b)}$. Demuestra que la retro propagación da como resultado una convolución mientras se giran los pesos. La rotación de los pesos se deriva de un error delta en la red neuronal

convolucional.

$$\frac{\partial E}{\partial i_{ij}^{l-1}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} \frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} w_{ab} \quad (2.2)$$

Ecuación 2.2 El error de propagación hacia atrás para la capa anterior se calcula usando la regla de cadena. Esta ecuación es similar a la definición de convolución, pero los pesos se rotan.

En CNN, pequeñas porciones de la imagen (llamadas campos receptivos locales) se tratan como entradas a la capa más baja de la estructura jerárquica. Una de las características más importantes de las CNN es que su arquitectura compleja proporciona un nivel de invariancia al cambio, la escala y la rotación, ya que el campo receptivo local permite que las neuronas o unidades de procesamiento accedan a características elementales, como bordes orientados o esquinas. Esta red se compone principalmente de neuronas con pesos y sesgos que se pueden aprender, formando la red convolucional.

La red neuronal convolucional (CNN) con estructuras profundas ha obtenido un enorme éxito en la clasificación de texto / no texto, detección humana, detección de oído, etc. En comparación con las redes neuronales superficiales tradicionales, tiene tres ventajas importantes: interacción escasa, intercambio de parámetros y equivalencia Ha mostrado ganancias significativas sobre los clasificadores de vanguardia, como la regresión logística, la máquina de aprendizaje extremo, la máquina de vectores de soporte y sus variantes, el clasificador de regresión lineal, etc. Una CNN típica incluirá una capa de convolución, una capa de activación no lineal y una capa de agrupación [Zhang et al., 2019].

Algunos usos prácticos validados de las CNN son:

- **Detección de actividad de voz en tiempo real** [Sehgal and Kehtarnavaz, 2018]: Las CNN procesan matrices como entradas, predominantemente imágenes, con sus capas ocultas que realizan funciones de convolución y agrupación junto con una capa totalmente conectada similar a una red neuronal de retropropagación convencional. Las capas de convolución son capaces de extraer información local de la imagen / matriz de entrada a través de los núcleos aprendibles ponderados con activaciones no lineales. Estos núcleos se replican en todo el espacio de entrada. Después de cada pasada, cada capa de convolución genera un mapa de características. Las capas de convolución están entrenadas para activar los mapas de características cuando se observan patrones de interés en la entrada. Estos mapas de características activadas se usan para sacar submuestras para reducir su resolución utilizando la agrupación máxima o convolución con zancadas

más largas, y luego se introducen en la siguiente capa de convolución. Las capas totalmente conectadas se utilizan para combinar la salida de la capa de convolución final y, por lo tanto, para clasificar la entrada general utilizando una capa de salida no lineal. La capa de salida en nuestro caso se considera una capa softmax que refleja las probabilidades asociadas con las dos clases correspondientes a ruido puro o solo ruido y voz + ruido o voz en ruido.

La figura 2.8 muestra como la red neuronal es estructurada para la detección de actividad de voz.

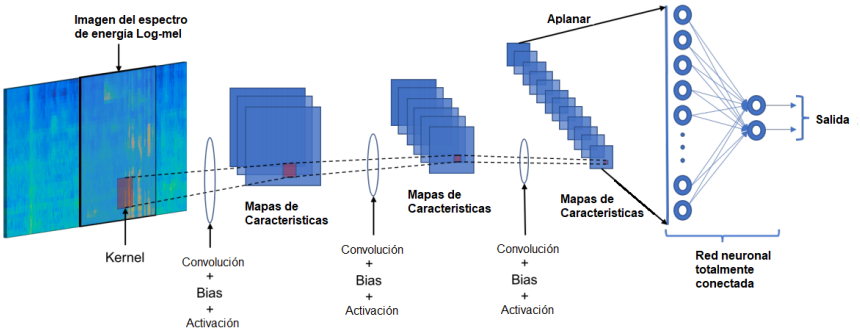


Figura 2.8: Ilustración del VAD (Siglas de Voice Activity Detection) basado en CNN desarrollado: La imagen del espectro de energía de log-mel se alimenta a las capas de convolución de CNN. La salida de la capa de convolución final se aplana en un vector y se alimenta a una capa completamente conectada. Finalmente, la salida de la capa totalmente conectada se alimenta a una capa softmax [Sehgal and Kehtarnavaz, 2018].

- **Reconocimiento Facial:** El requisito para una identificación personal confiable en el control de acceso computarizado ha resultado en un mayor interés en la biometría. Las caras representan estímulos visuales complejos multidimensionales significativos y es difícil desarrollar un modelo computacional para el reconocimiento facial [Lawrence et al., 1997].

  Las redes neuronales convolucionales (CNN) incorporan restricciones y logran cierto grado de cambio y deformación invariante usando tres ideas: campos receptivos locales, pesos compartidos y submuestreo espacial. El uso de pesos compartidos también reduce el número de parámetros en el sistema que ayuda a la generalización. Las redes convolucionales se han aplicado con éxito al reconocimiento de caracteres [Lawrence et al., 1997].

- **Clasificación de imágenes:** En CNN, los patrones de conectividad neuronal están inspirados en la corteza visual de los mamíferos. CNN mostró en primer lugar su capacidad sobresaliente en el reconocimiento de objetos en la competencia ImageNet. Más tarde, la CNN se ha aplicado ampliamente para analizar imágenes médicas, como segmentación de coroides, detección de anomalías, clasificación de núcleos de carcinoma, discriminación de quistes solitarios, análisis de datos espectroscópicos vibracionales, etc. [Zhang et al., 2019]

2.3.5. Redes neuronales recurrentes

Las redes neuronales recurrentes (RNNs) son un modelo poderoso para los datos secuenciales. Los métodos de entrenamiento de extremo a extremo, como la Clasificación Temporal de Connectionist, hacen posible entrenar a RNNs para problemas de etiquetado de secuencias donde la alineación entre entrada y salida es desconocido. La combinación de estos métodos con la arquitectura de memoria a corto plazo RNN ha demostrado ser particularmente fructífera, proporcionando resultados de última generación en el reconocimiento de escritura a mano cursiva [Graves et al.,].

Cuando la retropropagación fue introducida por primera vez, su uso más emocionante fue para el entrenamiento de redes neuronales recurrentes (RNN). Para las tareas que implican entradas secuenciales, como el habla y el lenguaje, a menudo es mejor usar RNNs (Figura 2.11). Los RNN procesan una secuencia de entrada un elemento a la vez, manteniendo en sus unidades ocultas un ”vector de estado”que contiene implícitamente información sobre la historia de todos los elementos pasados de la secuencia. Cuando consideramos las salidas de las unidades ocultas en diferentes etapas de tiempo discretas como si fueran las salidas de diferentes neuronas en una red multicapa profunda (Figura 2.11 derecha), queda claro cómo podemos aplicar la retropropagación para entrenar las RNNs.

Los RNN son sistemas dinámicos muy potentes, pero entrenarlos ha demostrado ser problemático porque los gradientes retropropagados crecen o se contraen en cada paso del tiempo, por lo que durante muchos pasos de tiempo normalmente explotan o desaparecen.

Gracias a los avances en su arquitectura y las formas de formación de los mismos, se ha encontrado que las RNN son muy buenas para predecir el siguiente carácter en el texto o la siguiente palabra en una secuencia, pero también pueden usarse para tareas más complejas. Por ejemplo, después de leer una oración en inglés una palabra a la vez, se puede entrenar una red çodificadoraïnglesa para que el vector de estado final de sus unidades ocultas sea una buena representación del pensamiento expresado por la oración. Este vector de pensamiento puede utilizarse entonces como el estado oculto inicial de (o como entrada adicional a) una red de decodificación francesa formada conjuntamente, que genera una distribución de probabilidad para la primera palabra de la

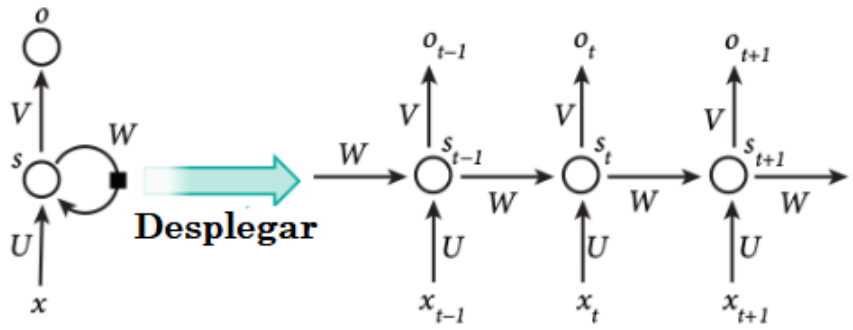


Figura 2.9: Una red neuronal recurrente y el despliegue en el tiempo del cómputo involucrado en su cómputo directo [LeCun et al., 2015].

traducción francesa. Si se elige una primera palabra particular de esta distribución y se proporciona como entrada a la red decodificadora, entonces emitirá una distribución de probabilidad para la segunda palabra de la traducción y así sucesivamente hasta que se elija una parada completa. En general, este proceso genera secuencias de palabras en francés según una distribución de probabilidad que depende de la oración en inglés. Esta manera bastante ingenua de realizar la traducción automática se ha convertido rápidamente en competitiva con el estado de la técnica, y esto plantea serias dudas acerca de si la comprensión de una oración requiere algo como las expresiones simbólicas internas que se manipulan mediante reglas de inferencia. Es más compatible con la opinión de que el razonamiento cotidiano implica muchas analogías simultáneas.

Las neuronas artificiales (por ejemplo, unidades ocultas agrupadas bajo el nodo s con valores st en el tiempo t) obtienen entradas de otras neuronas en pasos de tiempo anteriores (esto se representa con el cuadrado negro, que representa un retardo de un paso de tiempo, a la izquierda) . De esta manera, una red neuronal recurrente puede mapear una secuencia de entrada con elementos xt en una secuencia de salida con elementos ot, con cada ot dependiendo de todo xt'anterior (para $t' \leq t$). Los mismos parámetros (matrices U, V, W) se utilizan en cada paso de tiempo. Son posibles muchas otras arquitecturas, incluyendo una variante en la que la red puede generar una secuencia de salidas (por ejemplo, palabras), cada una de las cuales se utiliza como entradas para el siguiente paso de tiempo. El algoritmo de retropropagación puede aplicarse directamente al gráfico computacional de la red desplegada a la derecha para calcular la derivada de un error total (por ejemplo, la log-probabilidad de generar la secuencia derecha de salidas) con respecto a todos los estados st y todos los parámetros.

Algunos usos reconocidos de las RNN son:

- **Datos Secuenciales:** Como la mayoría de las redes neuronales, la red neuronal recurrente no es algo nuevo. Sus nodos se conectan direccionalmente entre sí en un bucle. Este estado interno puede mostrar el comportamiento dinámico de temporización de la red. A diferencia de las redes neuronales de avance, RNN utiliza su memoria interna para manejar cualquier secuencia de temporización de entrada, lo que facilita el manejo del reconocimiento de escritura y el reconocimiento de voz. [Shi et al., 2017].

Los RNN pueden aprender programas que combinan el procesamiento de información secuencial y paralela de una manera natural y eficiente, explotando el paralelismo masivo visto como crucial para mantener la rápida disminución del costo de cómputo observado en los últimos 75 años. [Schmidhuber, 2015].

Para usar de manera adecuada la técnica de deep learning se hace necesario tener conceptos claves que influyen en el flujo de la información, la manera en que serán almacenados los datos y la complejidad computacional que traen consigo, para ello se usará la definición de big data:

2.4. Big Data

El big data es un concepto abstracto. Aparte de las masas de datos, también tiene algunas otras características, que determinan la diferencia entre sí y ”datos masivos.º ”datos muy grandes”.

En la actualidad, aunque la importancia de big data ha sido generalmente reconocida, la gente todavía tiene diferentes opiniones sobre su definición. En general, big data significa los conjuntos de datos que no podrían ser percibidos, adquiridos, administrados y procesados por las herramientas tradicionales de TI y software / hardware dentro de un tiempo tolerable. Debido a las diferentes preocupaciones, las empresas científicas y tecnológicas, investigadores, analistas de datos, y los técnicos tienen diferentes definiciones de los grandes datos. Las siguientes definiciones pueden ayudarnos a comprender mejor las profundas connotaciones sociales, económicas y tecnológicas de los grandes datos [Chen et al., 2014a].

En 2010, Apache Hadoop definió big data como çonjuntos de datos que no podían ser capturados, administrados y procesados por computadoras en general dentro de un ámbito aceptable ”.

De esta definición, en mayo de 2011, McKinsey & Company, una agencia global de consultoría anunció Big Data como la próxima frontera para la innovación, la competencia y la productividad. Datos grandes significarán conjuntos

de datos que no podrían ser adquiridos, almacenados y administrados por el software clásico de base de datos. Esta definición incluye dos connotaciones: Primero, los volúmenes de los conjuntos de datos que se ajustan al estándar de los grandes datos están cambiando, y pueden crecer con el tiempo o con los avances tecnológicos; En segundo lugar, los volúmenes de conjuntos de datos que se ajustan al estándar de datos grandes en diferentes aplicaciones difieren entre sí. En la actualidad, los grandes datos varían generalmente de varios TB a varios PB. Desde la definición de McKinsey & Company, se puede ver que el volumen de un conjunto de datos no es el único criterio para datos grandes. La escala de datos cada vez más creciente y su gestión que no podría ser manejada por las tecnologías de base de datos tradicionales son las dos características clave siguientes [Chen et al., 2014a].

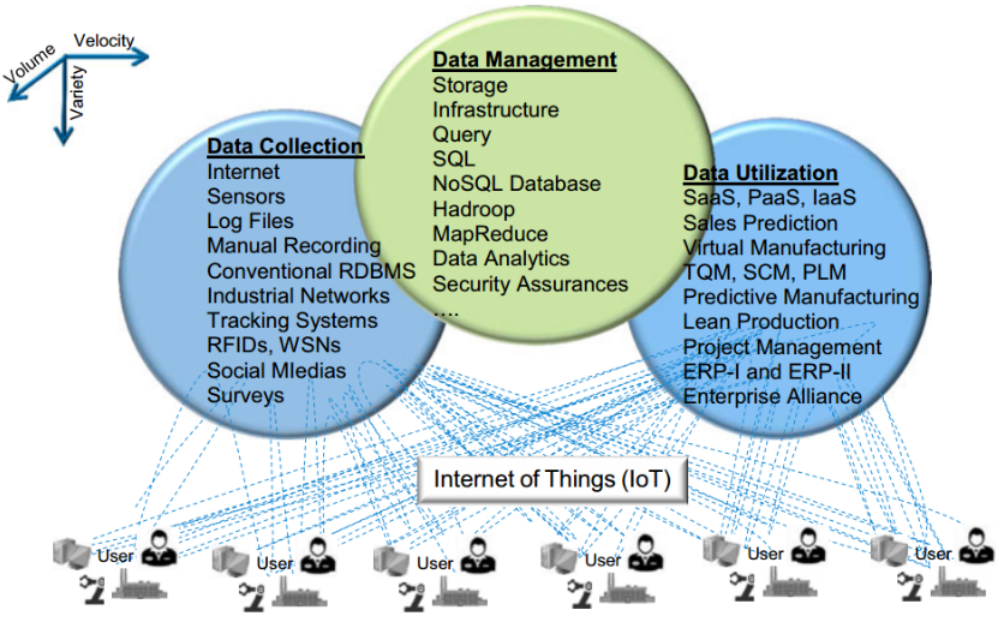


Figura 2.10: IoT, Cloud y Big Data [Chen et al., 2014a].

Además de los recursos computaciones con lo que ya se puede contar en la nube, los nuevos dispositivos IoT disponen de hardware con el cual se puede trabajar y delegar funciones, evitando así los costes adicionales, tanto económicos como computacionales, de llevar la totalidad de la carga de trabajo a la nube, para ello se usará el concepto de fog computing:

2.5. Fog Computing

Fog Computing extiende el paradigma de Cloud Computing al borde de la red. Mientras que Fog y Cloud usan los mismos recursos (redes, computación y almacenamiento), y comparten muchos de los mismos mecanismos y atributos (virtualización, tenencia múltiple), la extensión no es trivial, ya que existen algunas diferencias fundamentales que son la razón de ser de Fog Computing. La visión de Fog fue concebida para abordar aplicaciones y servicios que no se ajustan bien al paradigma de la nube. Incluyen:

- Aplicaciones que requieren una latencia muy baja y predecible: la nube libera al usuario de muchos detalles de implementación, incluido el conocimiento preciso de dónde tiene lugar el cálculo o el almacenamiento. Esta libertad de elección, bienvenida en muchas circunstancias, se convierte en una responsabilidad cuando la latencia es premium (juegos, videoconferencias).
- Aplicaciones geo distribuidas (monitoreo de tuberías, redes de sensores para monitorear el ambiente).
- Aplicaciones móviles rápidas (vehículo conectado inteligente, riel conectado).
- Sistemas de control distribuido a gran escala (red inteligente, ferrocarril conectado, sistemas inteligentes de semáforos).

La arquitectura cloud y la arquitectura Fog son compatibles entre sí y de acuerdo al caso de estudio no se recomienda reemplazar completamente la una o la otra. En la imagen [X] se muestra una arquitectura que contiene partes Cloud y partes Fog, diseñada específicamente para IoT.

Fog Computing extiende el paradigma de Cloud Computing al borde de la red, permitiendo así una nueva generación de aplicaciones y servicios. Las características definitorias de fog computing son:

- Baja latencia y conocimiento de la ubicación
- Distribución geográfica amplia
- Movilidad
- Gran cantidad de nodos
- Función predominante del acceso inalámbrico
- Fuerte presencia de aplicaciones de transmisión y en tiempo real
- Heterogeneidad [Bonomi et al., 2012]

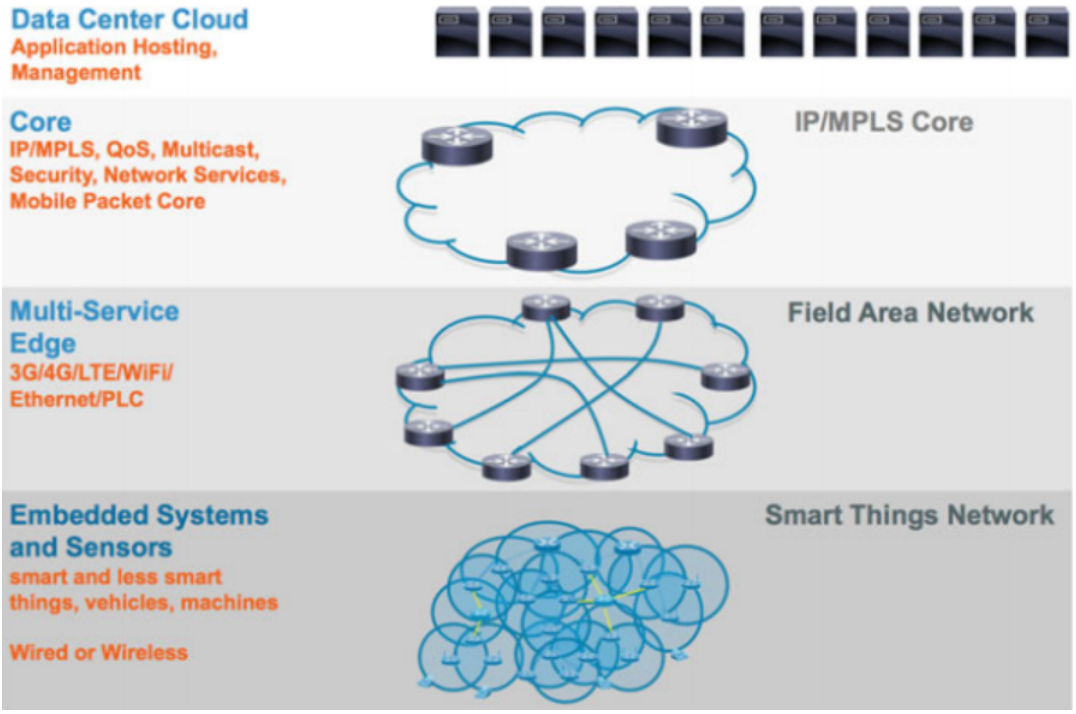


Figura 2.11: Infraestructura Fog distribuida para IoT/IoE [Bonomi et al., 2012]

Cuadro 2.1: Comparación MQTT y CoAP

Propiedad	MQTT	CoAP
Capa de Transporte	Principalmente TCP	Principalmente UDP, TCP puede ser usado pero no es usual
Arquitectura Soportada	Publicar-Suscribir	Basada en REST request / response. Observador de recurso (Publicar-Suscribir)
Estructura de cabecera y payload	Cabecera de 2 bytes. Payload tratado como un objeto binario largo.	Cabecera de 4 bytes + TLV (Type Length Value). Soporta codificación del payload cómo XML.
Tiempo de respuesta del timeout	Configurable. El tiempo de timeout aumenta a medida que se hacen re-intentos.	Por defecto 2 segundos. Puede ser configurado.
Máximas retransmisiones	Configurable.	4 y 0 para multicast.
Soporte Multicast	Distribución de mensajes uno a muchos es inherentemente soportada a través de publicar-suscribir	En el modo 'Observador de recurso' puede ser usado para mensajes de uno a muchos. El soporte para multicast dedicado existe pero no con retransmisión.
Confiabilidad	Con 3 niveles de QoS.	Fiabilidad simple de retransmisión de parada y espera con retroceso exponencial
Áreas de aplicación típica	Tipo de aplicación de mensajería en tiempo real basada en temas que utiliza pub / sub que requiere una conexión persistente con el servidor. El Broker de mensajes es responsable de enlazar los sensores con el resto de la Web.	Aplicaciones con sensores (posiblemente con un ciclo de suspensión definido) que requieren ejecutar servicios web RESTful para lograr una conectividad directa a la Web (u opcionalmente a algunos G / W) con métodos similares a HTTP y URI utilizando uno o ambos recursos de solicitud / respuesta u observación (variante del modo pub / sub) en tiempo real. Ideal para aplicaciones que requieren una fácil integración con la Web basada en HTTP. Adecuado para aplicaciones como energía inteligente, automatización de edificios, etc.

Capítulo 3

Descripción detallada del proceso

3.1. Materiales y métodos

3.1.1. *Benchmarking* a proveedores de servicios en la nube

El primer factor a considerar consistió en conocer las arquitecturas actualmente utilizadas en los servicios de nube más conocidos (Amazon Web Services, Microsoft Azure y Google Cloud Platform) en cuanto al tratamiento de datos de dispositivos IoT. Se encontró que hay similitud en la manera en que las nubes reciben, almacenan y permiten el análisis de los datos. En el trabajo realizado se concluyó que los tres factores más influyentes en la arquitectura son:

- **Transmisión:** La transmisión de los datos desde los dispositivos IoT sobre las mediciones realizadas y las predicciones iniciales que tienen orígen en los dispositivos del hogar hacia la nube se realiza a través del protocolo MQTT, fue una constante que cada proveedor de servicios en la nube ofrece un servicio de Broker MQTT por donde pasarán los datos de manera segura. En este punto es importante tener en cuenta que el servicio MQTT funciona bajo el modelo de publicar/suscribir, en donde un broker coordina la recepción de mensajes a través de temas y los reparte a aquellos que estén suscritos. La nube ofrece el broker intermedio que actúa como coordinador y a través de su SDK se transmiten datos y se suscribe como interesado en los mismos.
- **Almacenamiento:** Cuando los datos ya se encuentran en la nube, deben ser almacenados para su posterior análisis y trazabilidad, en este caso se requiere una base de datos no relacional que permita guardar datos no estructurados que se ajusta a la naturaleza de transmisión de dispositivos IoT. A pesar de que cada proveedor de servicios nube tiene sus propias

plataformas de almacenamiento se optó por utilizar un servicio común para que la solución no quede dependiente de un proveedor de servicios.

- **Análisis:** Para el análisis de datos se decide utilizar la plataforma de código abierto de extremo a extremo para el aprendizaje automático tensorflow como core principal de la analitica de datos basados en revisiones bibliográficas que constatan que en el campo de deep learning obtiene el balance entre rendimiento y complejidad de la solución llegando a resultados con altos índices de precisión de acuerdo al modelo construido. Además su versatilidad permite que sea ejecutado en varios entornos por lo que además de realizar análisis en la nube sin restricciones de proveedor, teniendo como único requisito un servidor preferiblemente con sistema operativo linux, se propone migrar parte del mismo a los dispositivos IoT haciendo uso de Fog Computing comprobando la capacidad computacional que los equipos de menor capacidad en su hardware pueden ofrecer.

Para cada plataforma se verificaron los costos de implementación de los servicios requeridos bajo cantidades similares de recursos computacionales. Una de las conclusiones para la toma de la decisión de la nube a adoptar era el balance entre el rendimiento y el costo que ésta pudiese ofertar. Para cada necesidad de la arquitectura se encontraron los siguientes servicios:

- Transmisión: Para la transmisión de datos se encontraron los servicios descritos en la tabla 3.1.
- Almacenamiento: Para el almacenamiento de kis datos se encontraron los servicios descritos en la tabla 3.2.
- Analítica: Para el análisis de los datos se encontraron los servicios descritos en la tabla 3.3. Es importante aclarar que en general no se buscaron los servicios de nube que ofrecen analítica ya que estos contienen sus propias adiciones y la idea de la arquitectura es que sea independiente del proveedor nube.

Para mayor información sobre los servicios, su descripción detallada, sus enlaces a la documentación oficial consultar el anexo [X] Benchmarking IoT.xlsx

Dado el estudio de benchmarking de proveedores de servicios en la nube el elegido para realizar la ejecución en este proyecto es Amazon Web Services, dejando claro que bajo las otras nubes consultadas la arquitectura a presentar continúa teniendo el mismo grado de validez ya que los servicios prestados en transmisión, almacenamiento y análisis se asemejan en cualquiera de los entornos y sólo cambia la configuración técnica de los servicios totalmente documentada en cada proveedor.

3.1.2. Distribución de los Datos analizados

Para el presente proyecto se utilizarán sensores node que arrojan datos en texto plano, además de imágenes tomadas a través de una cámara de 5mpx para la raspberry PI.En la tabla 3.4 se describe el dataset de los datos que serán almacenados y transmitidos a través de la arquitectura nube.

3.1.3. Dispositivos Físicos Utilizados

Durante la ejecución del proyecto se utilizaron una serie de dispositivos de Hardware que complementan la arquitectura completa del proyecto, en la tabla 3.5 se nombra el dispositivo utilizado, la versión actual y la función que cumple dentro de la arquitectura de analítica de datos IoT.

3.1.4. Servicios nube

Basados en el análisis de servicios mostrado en el punto 3.1.1 se define que los servicios nube a ser utilizados son los mostrados en la tabla 3.6. Estos servicios son propios de la nube de Amazon Web Services pero se deja claridad que las demás nubes consultadas cuentan con servicios que cumplen con la misma función con algunos detalles técnicos a ser configurados de acuerdo a la necesidad.

3.1.5. Aplicaciones de algoritmos de Deep learning para el hogar conectado

Los hogares cuentan actualmente con varios dispositivos IoT que sensan datos relevantes del entorno, razón por la cual la aplicación de técnicas que puedan sacar información basada en los datos obtenidos es aplicable. Algunos ejemplos de aplicación de técnicas de machine learning en el hogar conectado son:

- **Recomendaciones de entretenimiento:** Basado en los datos obtenidos de los dispositivos de entretenimiento (SmartTV, Consolas de Videojuegos, etc.) sobre los canales más visitados, los horarios habituales de entretenimiento, las preferencias de juegos, entre otros, es posible comunicar al usuario final información como cuando un programa que probablemente sea de su interés esté por iniciar, un juego de video esté por ser estrenado o una canción de su preferencia esté sonando en una cadena radial.
- **Seguridad del hogar:** Basado en los datos registrados en cámaras e interacción con los diferentes sensores del hogar es posible identificar si se intenta irrumpir la seguridad del hogar.

- **Prevención y detección de daños:** De acuerdo a la información obtenida por los dispositivos IoT se puede identificar si algún elemento del hogar se encuentra próximo a reparación, mantenimiento o dejó de funcionar correctamente.
- **Cuidado infantil:** Teniendo datos de todo lo que sucede en el hogar en un momento de tiempo específico es posible identificar si algún niño en la casa se encuentra en una situación posible de riesgo, alertando a sus mayores sobre lo sucedido.
- **Caracterización de la escena en la cocina:** En la cocina es posible obtener datos que permitan identificar cuando una persona está cocinando, además dada la identificación de cada alimento, el plato probable a ser preparado, los lugares de la cocina más usados entre otros. En este proyecto se realizó la prueba de validación de la arquitectura caracterizando los alimentos usados, colocándolos uno a uno en el misc and place, clasificando cada uno de ellos y reuniendo la información para posteriormente sabes qué tipo de plato se está preparando

Dependiendo de la naturaleza de los datos obtenidos, es posible usar diferentes técnicas de machine learning para obtener información relevantes y tomar decisiones basadas en datos. En la tabla 3.7 se especifican algunos casos de uso específico según el tipo de dato recolectado.

Para la presente arquitectura, dada la información recolectada en los textos científicos y en las pruebas realizadas, se definió utilizar las redes neuronales convolucionales (CNN) para la clasificación de imágenes capturadas desde la cámara de la raspberry PI, en el capitulo de resultados se mostrarán algunas comparaciones de rendimiento y precisión de los modelos probados bajo diferentes técnicas. Para el caso de los datos de texto subidos a la nube y almacenados en el motor de base de datos no relacional mongoDB, para el análisis y clasificación de la información se utilizaron máquinas de soporte vectorial (SVM).

3.1.6. Diseño de la solución

Los componentes generales que hacen parte de la arquitectura computacional propuesta como solución del problema se muestran en la figura 3.1.

En la figura 3.1 se describen todos los componentes que hacen parte de la arquitectura, en esta se muestra de manera general la disposición de los servicios requeridos. Los servicios de la nube fueron validados previamente en las principales plataformas (Google Cloud Platform, Microsoft Azure y Amazon Web Services) y todas ellas soportan, bajo diferentes nombres, la arquitectura propuesta con sus servicios requeridos.

En el desarrollo de este proyecto y su prueba de validación se eligió la plataforma nube Amazon Web Services dado el resultado del benchmarking realizado

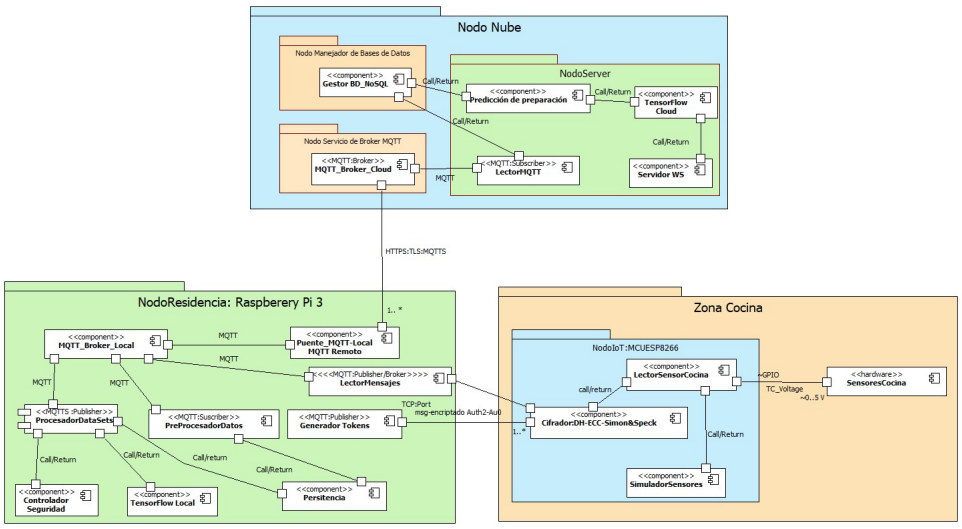


Figura 3.1: Arquitectura propuesta

en el que se evaluó los costes económicos de las plataformas y los servicios brindados (Ver Sección 3.1.1), en la 3.2 se puede evidenciar la arquitectura con los servicios de Amazon elegidos para cada labor.

La arquitectura se divide en dos componentes principales:

- **Componentes de hogar:** La primera parte de la arquitectura la forman los elementos que van directamente en la red local del hogar, a continuación se describe cada uno de ellos:
  - **Node:** Dispositivos que reciben los datos de los sensores y los entregan con doble cifrado al Broker instalado en la Raspberry.
  - **Raspberry:** A nivel de hogar este componente será el principal dentro de la arquitectura y tendrá una serie de funcionalidades distintas que se describen a continuación:
    - **Broker MQTT:** Recibir las peticiones de todos los sensores del hogar a través del protocolo MQTT con doble factor de cifrado. El broker MQTT Instalado es Mosca.
    - **FOG Computing:** Análisis de imágenes tomadas a través de la cámara de la Raspberry, clasificarlas en categorías de tipo de alimento previamente definidas en el modelo entrenado. Uso del framework de analitica de datos de TensorFlow.
    - **Publisher MQTT a la Nube:** Encargado de publicar al Broker MQTT de Amazon Web Services en el servicio AWS IoT Core.

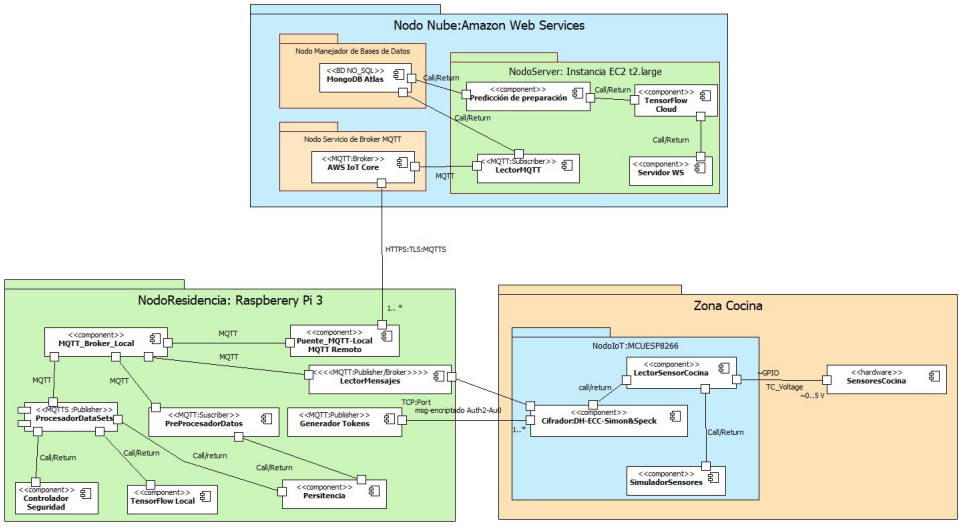


Figura 3.2: Arquitectura propuesta con los servicios de AWS

En las figuras 3.3 y 3.4 se pueden apreciar con mayor nitidez los componentes mencionados.

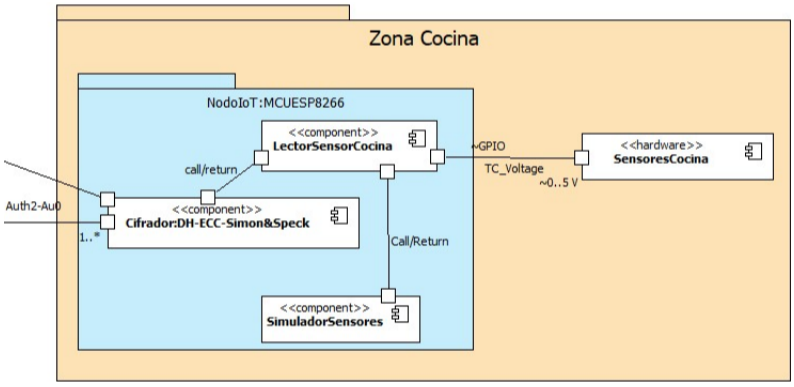


Figura 3.3: Nodo cocina en la arquitectura propuesta

- **Componentes de la Nube:** La segunda parte de la arquitectura la conforman los servicios de la nube que permiten recibir la información, almacenarla y finalmente analizarla. Los servicios y su respectivo uso dentro de la arquitectura se encuentran descritos a continuación:
    - **AWS IoT:** Broker MQTT que permite la recepción de los mensajes por parte del hogar y la entrega de los mismos al servidor. Todo el

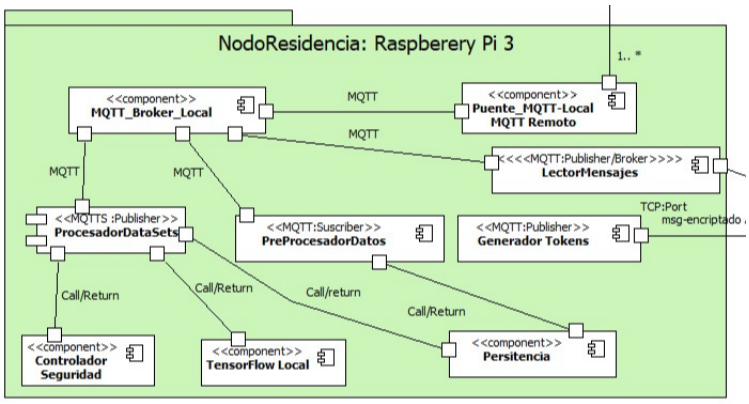


Figura 3.4: Nodo residencia en la arquitectura propuesta

funcionamiento de este broker se basa en el modelo publish/subscribe.

- **MongoDB Atlas:** Servicio de MongoDB que permite tener un cluster de bases de datos no relacionales. El modelo no relacional se ajusta de mejor manera a lo que posteriormente será el análisis de los datos. Este servicio se contrata de manera directa con MongoDB, sin embargo es ejecutado sobre el servicio de Nube de Amazon Web Services.
- **VPS Linux:** Servicio de servidor virtual privado en la nube de Amazon Web Services, el servidor cumplira con las siguientes funciones
  - **Subscriber MQTT:** El servidor se encontrará suscrito a los mensajes recibidos en el broker MQTT de AWS IoT.
  - **Almacenamiento en la base de datos:** Luego de tener los mensajes en el servidor, este procederá a almacenarlos en la base de datos MongoDB.
  - **Análisis de Datos:** A través del framework de TensorFlow el servidor tendrá la tarea de ejecutar scripts que tomen la información de la base de datos y realice una labor de clasificación acerca de la tarea que se está ejecutando en la cocina.

En la figura 3.5 se puede apreciar con mayor nitidez el nodo de la nube descrito.

En resumen en la arquitectura propuesta, iniciando por las labores de hogar, se tiene que la raspberry PI será la encargada de recibir los resultados de las mediciones de los sensores instalados en la cocina a través del protocolo MQTT siendo broker de los dispositivos hogar, posteriormente se realiza *fog computing* en el mismo dispositivo, en este caso de estudio con la finalidad de clasificar las imágenes capturadas desde la cámara a los alimentos que se encuentren en

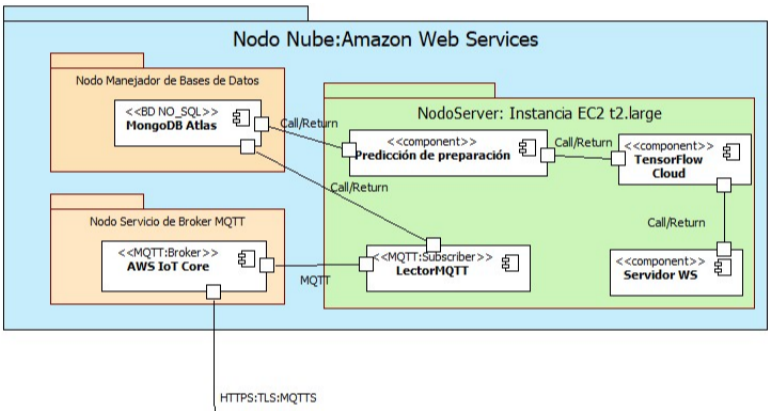


Figura 3.5: Nodo nube en la arquitectura propuesta con los servicios de AWS

el mise en place , luego de clasificarlas y agrupar los datos recibidos por los sensores publica al servicio *AWS IoT Core* situado en la nube de *Amazon Web Services* en un tema MQTT ya especificado por la arquitectura. En la figura 3.6 se puede observar la manera en que los datos fluyen a través de los distintos componentes de la arquitectura propuesta usando el protocolo MQTT.

Cuando el dato ya se encuentra en la nube, el VPS con SO Ubuntu tiene en ejecución un script que se encuentra suscrito al mismo tema de publicación de la Raspberry PI, por lo que recibe los datos del servicio de amazon y los almacena en la base de datos no relacional Mongo DB que se encuentra en el servicio nube Mongo DB Atlas. luego de tener una cantidad [X] de registros almacenados obtiene nuevamente los registros, esta vez de la base de datos, y los usa como datos de entrada a la segunda red neuronal de análisis de información en la que se pretende conocer la actividad que se está haciendo en la cocina.

3.1.7. Detalles de aplicación implementación y validación

Transmisión de los datos a través de MQTT con el servicio de AWS IoT Core

La primer parte implementada de la arquitectura fué la transmisión de los datos desde los dispositivos hogar a la nube haciendo uso del servicio AWS IoT Core. Como se mencionó en la arquitectura propuesta el dispositivo hogar encargado de la transmisión es la Raspberry PI. El lenguaje de programación utilizado para realizar el script de envío es Python. En la siguiente dirección URL de la plataforma github se encuentra el código fuente usado en el envío.

https://github.com/ccasta23/aws-tf-iot-analysis/tree/master/client-side

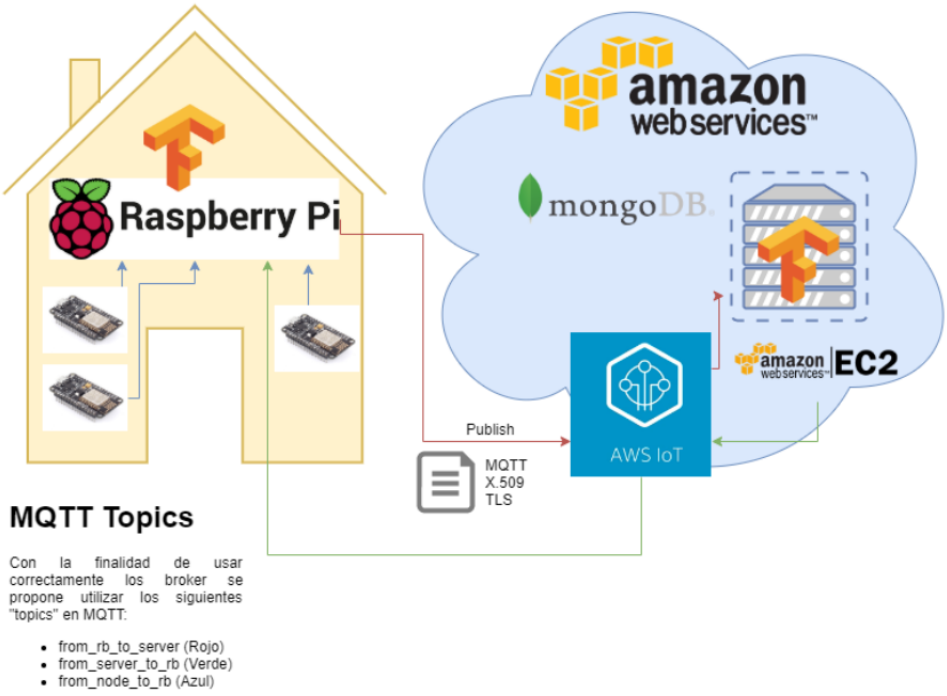


Figura 3.6: Flujo de datos de la arquitectura propuesta

Para publicar al servicio de AWS IoT Core a través del protocolo MQTT se debe hacer uso del paquete de transmisión entregado por Amazon, este se encuentra en diferentes lenguajes de programación; la elección en este caso fué Python pero se deja claridad que la arquitectura propuesta se puede implementar bajo cualquier lenguaje de programación soportado por amazon para la transmisión.

Con la finalidad de que el código realizado sirva para posteriores proyectos que deseen hacer uso de la arquitectura propuesta para envío de datos a través de MQTT a la nube de Amazon Web Services, se modifican los archivos fuente de publicación y se agrega un archivo de configuración similar al siguiente:

```
{
    "host": "YOUR_AWS_HOST",
    "rootCAPath": "certificados/root-CA.crt",
    "certificatePath": "certificados/YOUR_CERT",
    "privateKeyPath": "certificados/YOUR_PRIVATE_KET",
    "port": YOUR_PORT,
    "useWebsocket": false,
    "clientId": "YOUR_CLIENT_ID",
```

```
    "topic": "YOUR_TOPIC",
    "mode": "publish"
}
```

Con dicho archivo se abre la posibilidad para que cualquier dispositivo IoT que se encuentre en la arquitectura y pueda correr un script en Python se identifique y publique información ante la nube de AWS. Los datos solicitados en el archivo de configuración son únicos por cada dispositivo IoT que desee publicar y pueden ser encontrados en la consola de administración de Amazon Web Services en el servicio AWS IoT Core.

Almacenamiento de los datos obtenidos a través de MQTT desde el servicio de AWS IoT Core

Para almacenar los datos, como se mencionó anteriormente, se hace uso del servicio nube MongoDB Atlas creado por MongoDB. Este servicio de base de datos no relacional basado en documentos será el encargado de mantener la información de manera persistente.

El servicio de MongoDB atlas puede ser consultado en el siguiente enlace:

https://www.mongodb.com/cloud/atlas

Posteriormente se realiza la configuración del servidor *VPS* con sistema operativo *Ubuntu* que se encuentra suscrito al servicio *AWS IoT Core* en el tema específico en el que publica la *raspberry*. El código fuente de la aplicación encargada de recibir los mensajes de *AWS IoT Core* y almacenarlos en la base de datos *MongoDB* se encuentra en el siguiente enlace:

https://github.com/ccasta23/aws-tf-iot-analysis/tree/master/server-side

Con la finalidad de que el código realizado sirva para posteriores proyectos que deseen hacer uso de la arquitectura propuesta para recepción y almacenamiento de datos a través de *MQTT* y guardandolos en MongoDB Atlas, se modifican los archivos fuente de publicación y se agregan dos archivos de configuración, el primero de ellos haciendo alusión a los datos de suscripción a *MQTT*, como se puede ver a continuación:

```
1  {
2      "host": "YOUR_AWS_HOST",
3      "rootCAPath": "certificados/root-CA.crt",
4      "certificatePath": "certificados/YOUR_CERT",
5      "privateKeyPath": "certificados/YOUR_PRIVATE_KET",
6      "port": YOUR_PORT,
```

```
        "useWebsocket": false,
        "clientId": "YOUR_CLIENT_ID",
        "topic": "YOUR_TOPIC",
        "mode": "subscribe"
    }
```

Además se crea un archivo de configuración con los datos de conexión a la base de datos creada en MongoDB como el que se observa a continuación:

```
    {
        "DB_URI": "YOUR_MONGO_DB_URI"
    }
```

Con ambos archivos modificados con datos correctos se tendrá funcional en el servidor el *script* que recibe los datos vía suscripción *MQTT* y los almacena en la base de datos *MongoDB*.

Parámetros del modelo de análisis de datos

Para implementar la solución de análisis de datos, se usó la herramienta de *tensorflow*. Con la finalidad de obtener la solución más óptima en el proceso de analizar los datos y clasificarlos, es necesario definir una serie de parámetros a la herramienta para que así tenga un punto de partida que le permita hacer mejor el trabajo y pueda llegar a un modelo de clasificación más preciso y rápido tanto en su fase de entrenamiento como en la fase de clasificación.

A continuación se presenta una porción del código de entrenamiento de la red neuronal convolucional en donde se definen las capas, la función de optimización, la función de activación entre otros.

```

model = Sequential()

#Capa de entrada.
model.add(InputLayer(input_shape=(img_size_flat,)))
#Reformar la imagen, la convierte a matriz nuevamente.
model.add(Reshape(img_shape))

#Primera capa convolucional
model.add(Conv2D(kernel_size=5, strides=1, filters=16,
padding='same', activation='relu', name='layer_conv1'))
model.add(MaxPooling2D(pool_size=2, strides=2))

```

```
#Segunda capa convolucional
model.add(Conv2D(kernel_size=5, strides=1, filters=36,
padding='same', activation='relu', name='layer_conv2'))
model.add(MaxPooling2D(pool_size=2, strides=2))

#Aplanamiento de la imagen.
model.add(Flatten())
#Capa densa, se abstraen las características más relevantes.
model.add(Dense(128, activation='relu'))
#Capa de salida.
model.add(Dense(num_classes, activation='softmax'))

#Compilación del modelo
optimizer = Adam(lr=1e-3)
model.compile(optimizer=optimizer,
    loss='categorical_crossentropy',
metrics=['accuracy'])

#Entrenamiento del modelo
#Epochs: Número de veces que entrena.
#Batch_size: tamaño de lotes para entrenamiento.
model.fit(x=images, y=probabilities, epochs=1, batch_size=30)


limit_test_images = 40
test_images, test_tags, test_probabilities = load_data("test/",
num_classes, limit_test_images)
results = model.evaluate(x=test_images, y=test_probabilities)
print("{0}: {1:.2%}" .format(model.metrics_names[1],
    results[1]))

#Se usa la librería de Keras
name_file = 'modelo.keras'
#Se guarda el archivo en la carpeta raíz.
model.save(name_file)
#Se imprime la estructura del archivo, reporte.
model.summary()
```

Dentro de la parametrización de la herramienta encontramos las siguiente opciones:

- Tipos de Capas de un modelo de análisis de datos
- Ajuste de Hiper parámetros
- Funciones de activación
- Métodos de optimización

Fog Computing en la *Raspberry PI*

Con la finalidad de no delegar la totalidad de la labor a la nube, reducir tiempos de latencia y uso de red, utilizar gran parte de los recursos computacionales de cada dispositivo y tener una arquitectura de hogar con mayor movilidad se decide probar el uso del paradigma del *fog computing*.

Como eje central de la arquitectura del lado del hogar se encuentra la *Raspberry PI*, dispositivo encargado de centralizar la totalidad de los datos y capturar más a través de la cámara instalada. Dados los recursos computacionales con los que cuenta este dispositivo, que son altos para dispositivos del tamaño del mismo, se decide probar el rendimiento que tiene, ejecutando un análisis de imagen capturada por la cámara instalada. Para el análisis de la imagen se utiliza el *framework* de *TensorFlow* con *deep learning* haciendo uso de una red neuronal convolucional (CNN) previamente entrenada en un entorno de mayores recursos y trasladada a la *Raspberry* para que su labor se centre únicamente en el análisis y clasificación de la misma.

Las dos principales variables a evaluar son el rendimiento y la precisión. Como indicador de medición del rendimiento se tiene el tiempo de respuesta, este es un factor importante teniendo en cuenta que en un momento dado se pueden capturar un flujo de imágenes y cada una de ellas debe ser analizada y clasificada en periodos no superiores a los 5 segundos de la siguiente captura.

Por otro lado la precisión de la clasificación debe estar en rangos superiores al 70 %, ya que de cualquier otra manera, por eficiente que sea la clasificación no tendrá los resultados esperados si no se logra clasificar correctamente la imagen analizada.

Arquitectura de la redes neuronales convolucionales

En el *fog computing* se utilizó la técnica de *deep learning* para categorizar imágenes de algunos ingredientes de cocina. Es importante resaltar que el *dataset* usado se encontraba sesgado a un conjunto de datos pequeño con las siguientes categorías: huevos, mantequilla, arepas, chocolate, pan. A continuación se presenta la porción de código en donde se adicionan los parámetros, hiper parámetros, capas y función de activación:

```
1 # Las imágenes del dataset tienen dimensión de 128x128.
2 img_size = 128
3
4 # Las imágenes son almacenadas en arreglos de una dimensión de
    este tamańo.
```

```
5  img_size_flat = img_size * img_size
6
7  # Tupla con ancho y alto de las imágenes usadas para
       redimensionar los arreglos.
8  # Esto es usado para la imprimir las gráficas.
9  img_shape = (img_size, img_size)
10
11 # Tupla con el alto, ancho y profundidad usadas para
       redimensaionar los arreglos.
12 # Esto es usado para redimensionar en Keras
13 img_shape_full = (img_size, img_size, 1)
14
15 # Número de canales de color para las imágenes: 1 canal para
       escala de grises.
16 num_channels = 1
17
18 # Cantidad de categorías creadas para clasificar
19 num_classes = 5
20
21 # Comienza la construcción del modelo Keras Sequential.
22 model = Sequential()
23
24 # Agrega una capa de entrada que es similar a un feed_dict en
       TensorFlow.
25 # Tenga en cuenta que la forma de entrada debe ser una tupla
       que contenga el tamaño de la imagen.
26 model.add(InputLayer(input_shape=(img_size_flat,)))
27
28 # La entrada es una matriz aplanada con 784 elementos
       (img_size * img_size),
29 # pero las capas convolucionales esperan imágenes con forma
       (28, 28, 1), por tanto hacemos un reshape
30 model.add(Reshape(img_shape_full))
31
32 # Primera capa convolucional con ReLU-activation y max-pooling.
33 model.add(Conv2D(kernel_size=5, strides=1, filters=16,
       padding='same',activation='relu', name='layer_conv1'))
34 model.add(MaxPooling2D(pool_size=2, strides=2))
35
36 # Segunda capa convolucional con ReLU-activation y max-pooling.
37 #padding = same --> imagen de salida mismo tamaño de la entrada
38 model.add(Conv2D(kernel_size=5, strides=1, filters=36,
       padding='same',activation='relu', name='layer_conv2'))
39 model.add(MaxPooling2D(pool_size=2, strides=2))
40
41 # Aplanar la salida de 4 niveles de las capas convolucionales
42 # a 2-rank que se puede ingresar a una capa totalmente
       conectada
43 model.add(Flatten())
44
```

```
# Primera capa completamente conectada  con ReLU-activation.
model.add(Dense(128, activation='relu'))
# última capa totalmente conectada con activación de softmax
    para usar en la clasificación.
model.add(Dense(num_classes, activation='softmax'))

#Compilación del modelo
#función de coste, un optimizador y las métricas de rendimiento
model.compile(optimizer="adam",
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Luego de entrenar la red neuronal convolucional es posible exportar métricas y generar la arquitectura de la misma usando la herramienta *tensorboard*. Para el caso de estudio actual la arquitectura obtenida se muestra en la figura 3.7.

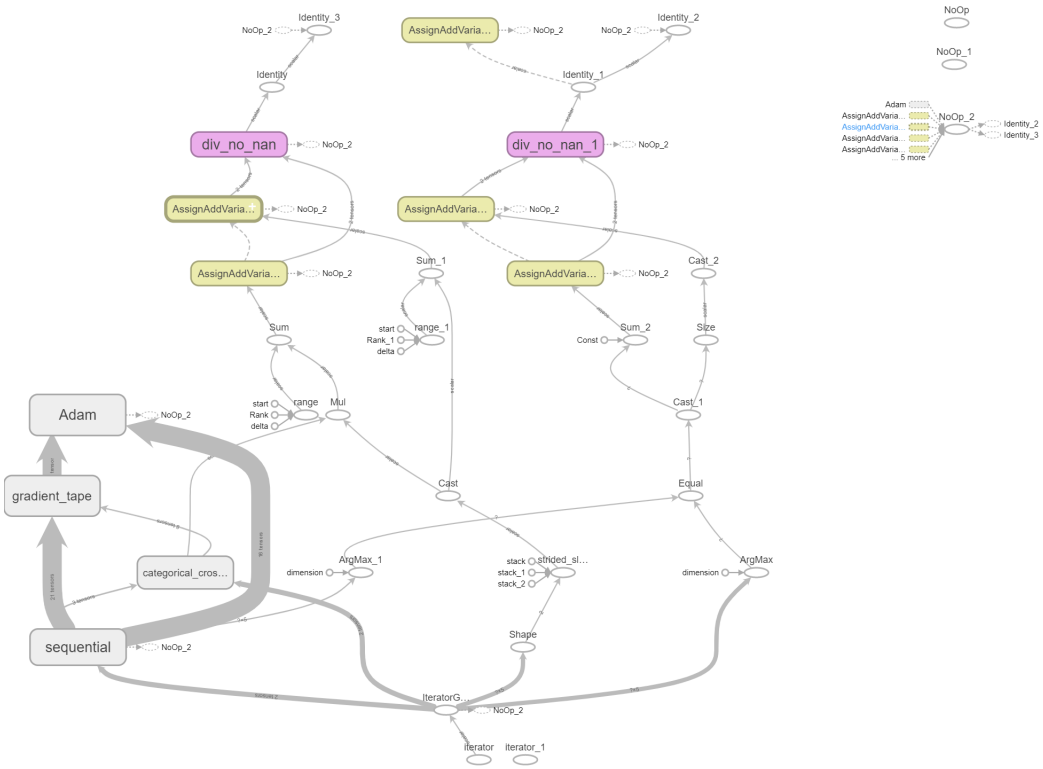


Figura 3.7: Arquitectura de la red neural convolucional

La mejor precisión del modelo (*accuracy*) lograda fue del 99.5 % con 30 *epoch* y con la función de activación *ReLU*. En la tabla 3.8 se puede apreciar diferentes *accuracy* obtenidos dependiendo de la configuración de la red neuronal.

3.1.8. Resumen

Dentro del proceso de diseño de la arquitectura se definen tres puntos focales: Análisis, transmisión y almacenamiento de los datos. La arquitectura propuesta aborda las tres temáticas y entrega diferentes opciones de servicios, proveedores y dispositivos que la hacen flexible y permite ser aplicada en diferentes contextos de negocio.

En cuanto a la transmisión de los datos se elige el protocolo de transmisión MQTT que brinda rapidez y seguridad en la transmisión. Las características y ventajas del protocolo pueden ser revisadas en la tabla 2.1. Este protocolo a pesar de que no es nuevo, es cada vez más utilizado en casos de uso que se solucionen aplicando *IoT* dado que trabaja bajo el modelo de publicación/suscripción que permite persistencia en la conexión y flujo constante de datos entre todas las partes implicadas.

En cuanto al análisis de la información se propone usar la totalidad de los recursos computacionales que conforman la arquitectura. Inicialmente se plantea que los dispositivos *IoT* además de sensar y transmitir los datos también tengan un análisis parcial de los mismos, esto permite reducir costos de transmisión y hacer que estos dispositivos operen sobre toda su capacidad. Todo el proceso de análisis previo permite que los datos lleguen procesados a la nube, y en ese espacio se puedan realizar nuevos análisis predictivos usando todas las capacidades computacionales que las instancias ofrecen. De acuerdo a lo propuesto en la presente arquitectura, los análisis realizados en ambos entornos pueden ser realizados bajo cualquier técnica de *machine learning* (*Deep learning, SVM, Clustering con K-Means*) siempre y cuando el sistema operativo sea basado en *linux*. Se recomienda revisar la tabla 3.7 para ver los usos más comunes de cada técnica.

Finalmente en el almacenamiento se decide dejar una base de datos no relacional que permite mayor flexibilidad dada la naturaleza de los datos recolectados y los resultados de los análisis. Esta elección hace que incorporar un nuevo dispositivo que transmita datos o un nuevo análisis que arroje información previamente no conocida, no necesite grandes cambios sobre el modelo de los documentos de bases de datos. El almacenamiento es la parte de la arquitectura que cuenta con mayor pluralidad de servicios que pueden ser utilizados, algunos de estos pueden ser consultados en la tabla 3.2.

Cuadro 3.1: Benchmarking servicios de transmisión de nube

Servicio	Costo	Plataforma
AWS IoT Core	**Conectividad** Precios del servicio de conectividad (por millón de minutos de conexión) N. Virginia 0,080 USD **Mensajería** Precios del servicio de mensajería (por millón de mensajes) N. Virginia Hasta mil millones de mensajes 1,00 USD Siguientes 4 mil millones de mensajes 0,80 USD Más de 5 mil millones de mensajes 0,70 USD **Registro y sombra de dispositivos** Precios del registro y las sombras de dispositivos (por millón de operaciones) N. Virginia 1,25 USD **Motor de reglas** Precios del servicio de motor de reglas (por millón de reglas disparadas/por millón de acciones ejecutadas) N. virginia Reglas disparadas 0,15 USD Acciones ejecutadas 0,15 USD	Amazon
Cloud IoT Core	**Google Cloud IoT Core Pricing** Volúmen Mensual Precio por MB Hasta 250 MB $0,00 Entre 250 MB y 250 GB $0,0045 Entre 250 GB y 5 TB $0,0020 5 TB o superior $0,00045	Google Cloud Platform
Azure IoT HUB	**Edición Precio por unidad (Al mes) Mensajes por dia por unidad** Gratis Gratis 8,000 S1 $50 400,000 S2 $500 6,000,000 S3 $5,000 300,000,000	Microsoft Azure

Cuadro 3.2: Benchmarking servicios de almacenamiento NoSQL de nube

Servicio	Costo	Plataforma
AWS S3	**Almacenamiento** Primeros 50 TB/mes $0.023 por GB $0.0125 por GB $0.004 por GB Siguientes 450 TB/mes $0.022 por GB $0.0125 por GB $0.004 por GB Más de 500 TB/mes $0.021 por GB $0.0125 por GB $0.004 por GB **Solicitudes** Solicitudes PUT, COPY, POST o LIST $0.005 por cada 1 000 solicitudes GET y el resto de solicitudes $0.004 por cada 10 000 solicitudes Solicitudes de eliminación Free **Transferencia de datos** Transferencia ENTRANTE de datos a Amazon S3 Todas las transferencias entrantes de datos $0.000 por GB Transferencia SALIENTE de datos de Amazon S3 a EE.UU. Este (Norte de Virginia) $0.010 por GB Transferencia SALIENTE de datos de Amazon S3 a Internet Primer GB/mes $0.000 por GB Hasta 10 TB/mes $0.090 por GB Siguientes 40 TB/mes$0.085 por GB	Amazon
Cloud BIG Table	**Google Cloud Big Table Pricing** Característica Precio Nodos $0,65 nodo/hora Almacenamiento en SSD $0,17 (GB al mes) Almacenamiento en HDD $0,026 (GB al mes) Entrada en red GRATIS Salida de red Se aplican tarifas de salida de Internet y entre regiones	Google Cloud Platform
Azure Cosmos DB	**Unidad Precio** Almacenamiento SSD (por GB) $0.25 GB/mes RU reservadas/segundo (por 100 RU, 400 RU como mínimo) $0.008/hora	Microsoft Azure
MongoDB Atlas	**Despliegue en la nube de AWS, Región N. Virginia, 5GB de almacenamiento**. $0.012/hora* Costo por transferencia de datos $0.01/GB En la misma región Precio estimado para el conjunto de réplicas de 3 nodos	Independiente del proveedor de nube

Cuadro 3.3: Benchmarking servicios de análisis de datos de nube

Servicio	Costo	Plataforma
TensorFlow en AWS	**Instancia EC2 del Marketplace** $0,003 por hora **Máquina EC2** T2 Nano $0.00 $0.006 $0.006/hr	Amazon
Cloud Machine Learning Engine	**Elemento EE. UU.** Clústeres de preparación Nivel básico 0,49 USD por hora Nivel estándar 4,90 USD por hora Nivel premium 36,75 USD por hora Configuración de clústeres personalizada 0,49 USD por hora y unidad de preparación del aprendizaje automático Nivel GPU básica 1,47 USD por hora **Solicitudes de predicción** Hasta 100 millones al mes 0,10 USD cada mil +0,40 USD por nodo y hora Más de 100 millones de solicitudes al mes 0,05 USD cada mil +0,40 USD por nodo y hora	Google Cloud Platform
Machine Learning services	**Precios de Administración de modelos** Desarrollo y pruebas Estándar S1 Estándar S2 Estándar S3 * Precio de nivel al mes $0 $50 $375 $2,500 **Características** Modelos administrados 20 100 1,000 10,000 Implementaciones administradas 2 10 100 1,000 Núcleos disponibles** 4 16 120 800	Microsoft Azure

Cuadro 3.4: Dataset

Dispositivo de Recolección	Descripción
Sensor de Temperatura	Dato de tipo numérico. Describe la temperatura del ambiente.
Sensor de Flujo	Dato de tipo numérico. Envía 0 en caso de que el flujo sea interrumpido y 1 en caso de que el flujo sea iniciado.
Sensores electrónicos. Electrodomésticos: Licuadora, estufa	▪ ¿Qué electrodoméstico?. Dato numérico. ▪ Fecha. Dato tipo fecha ▪ ¿Por cuánto tiempo?. Dato numérico
Sensores magnéticos	Dato de tipo numérico. Envía 0 en caso de que la gaveta sea cerrada y 1 en caso de que la gaveta sea abierta.
Cámara de Raspberry	▪ ¿Qué elementos se encuentran en el mise en place? ▪ ¿Qué elementos se toman de las gavetas?. Salida del primer modelo de Análisis de datos. Posterior a análisis de imagen ▪ ¿Qué elementos se toman del refrigerador?. Salida del primer modelo de Análisis de datos. Posterior a análisis de imagen

Cuadro 3.5: Dispositivos físicos usados

Dispositivo	Versión	Función dentro de la arquitectura
Raspberry PI 3	Model B	▪ Broker MQTT de los dispositivos internos de la cocina. ▪ Publisher MQTT hacia el servicio nube con los datos consolidados. ▪ Realiza análisis de imágenes tomadas desde la cámara para clasificación de alimentos usando tensorflow.
Node	MCU ESP8266	▪ Lectura de datos arrojados por los sensores. ▪ Publisher MQTT a la Raspberry.
Sensor magnético de Puertas - Ventanas	MC-38	▪ Sensar la apertura o cierre de las gavetas de la cocina.
Sensor de Corriente	ACS712	▪ Sensar el encendido o apagado de los electrodomésticos.
Sensor de Movimiento piroeléctrico	HC-SR501	▪ Sensar la detección del movimiento en la cocina.

Cuadro 3.6: Servicios Nube seleccionados

Servicio	Proveedor	Descripción de utilidad en la arquitectura
AWS IoT Core	Amazon Web Services.	Servicio de transmisión de datos desde dispositivos IoT a la nube a través del protocolo MQTT
Mongo DB Atlas	MongoDB	Servicio de almacenamiento de datos no estructurados haciendo uso del motor de base de datos no relacional MongoDB. Ejecuta sobre todas las nubes consultadas.
AWS EC2	Amazon Web Services.	VPS con sistema operativo Ubuntu 16.04 que sirve como gestor intermedio en recibir los datos siendo subscriber MQTT, recibiendo los datos arrojados por los dispositivos, almacenarlos en la base de datos MongoDB y posteriormente analizandolos haciendo uso de la librería Tensorflow.

Cuadro 3.7: Tipos de datos para algunas técnicas de Machine Learning

Técnica de Machine Learning	Usos recomendados	Tipos de dataset con los que mejor se comporta
Redes Neuronales Convolucionales (CNN)	▪ Reconocimiento de imágenes [Kalchbrenner et al., 2014, Moeskops et al., 2016] ▪ Reconocimiento de voz [Sehgal and Kehtarnavaz, 2018] ▪ Textos cortos [Dos Santos and Gatti, 2014, Severyn and Moschitti, 2015]	▪ Imágenes ▪ Matrices multidimensionales ▪ Textos cortos ▪ Audio
Redes Neuronales Recurrentes (RNN)	▪ Procesamiento de lenguaje natural [Sak et al., 2014] ▪ Predicción de texto [Mikolov et al., 2010]	▪ Texto
Máquinas de soporte Vectorial (SVM)	▪ Clasificación [Foody and Mathur, 2006] ▪ Reconocimiento de patrones [Guzmán et al., 2017]	▪ Imágenes ▪ Texto
Clustering con K-Means	▪ Particionar el dataset [Wagstaff et al., 2001]	▪ Texto

Cuadro 3.8: *Accuracy* de la red neuronal con dos capas convolucionales asignando diferentes parámetros

Parámetros	*Accuracy*
▪ Función de activación: *ReLU* ▪ Epoch: 30	99.5 %
▪ Función de activación: *Tanh* ▪ Epoch: 30	33.5 %
▪ Función de activación: *ReLU* ▪ Epoch: 10	76.3 %
▪ Función de activación: *Sigmoide* ▪ Epoch: 10	19.5 %
▪ Función de activación: *Elu* ▪ Epoch: 10	40.0 %

Capítulo 4

Análisis de Resultados

A continuación se presentan los resultados obtenidos bajo la prueba de validación realizada en el proyecto, iniciando por los resultados del fog computing en la raspberry PI hasta los servicios nube utilizados.

4.1. Resultados del Fog computing

Al iniciar las pruebas de análisis en la Raspberry y monitorear el uso de recursos de la misma, se concluye que inicialmente Tensowflow no está usando todos los núcleos que la CPU tiene disponibles y está desaprovechando recursos de Hardware que están preparados para ser usados.

Con la finalidad de aprovechar la totalidad de los recursos computacionales ofrecidos por la Raspberry PI se realizó una recompilación del core de tensorflow logrando que utilice los 4 núcleos de la CPU de manera paralela y además use toda la memoria RAM disponible en la ejecución del análisis de los datos.

Uno de los grandes inconvenientes encontrados es que iniciar el core de Tensorflow tarda aproximadamente 30 segundos en la Raspberry, por lo que inicialmente en cada ejecución de lectura y clasificación de la imagen se tenía que contar con el tiempo de subida del core sumado al tiempo de clasificación. Como solución al problema se propone dejar un script en ejecución que inicie el core de Tensorflow y periódicamente revise el contenido de una carpeta en donde encontrará las imágenes a clasificar.

Los recursos de la Raspberry son ocupados por completo en el momento de ejecución del script de clasificación y tarda aproximadamente 30 segundos en iniciar el core de Tensorflow. Los cuatro procesadores del dispositivo se ven copados en su totalidad (Ver figura 4.1) dejando claro que los únicos procesos adicionales a los del sistema operativo son el script en ejecución, la consola de análisis de recursos y la consola para toma de captura de pantalla.

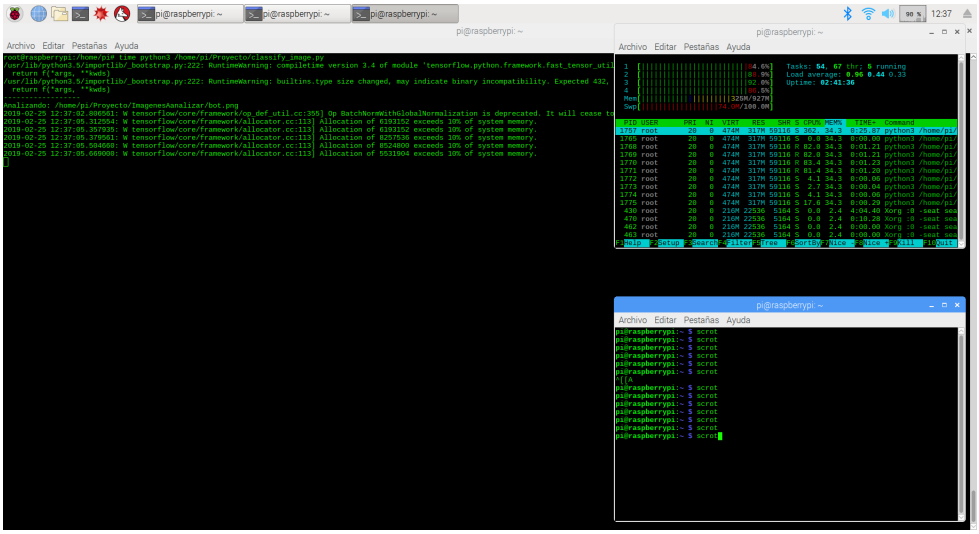


Figura 4.1: Tensorflow en Raspberry usando todos los recursos

En la primera prueba realizada se ejecutó un ejemplo del repositorio de ejercicios de TensorFlow que clasifica una imagen de acuerdo al repositorio de imágenes MNist. La prueba se ejecutó en varios entornos para probar el rendimiento y la precisión que cada dispositivo entregaba. A continuación se describen los dispositivos utilizados:

Laptop:

Las características de recurso de hardware disponibles en la máquina se encuentran descritas en la figura 4.2.

Figura 4.2: Recursos disponibles Laptop

Los resultados obtenidos del análisis de los datos en cuanto a rendimiento y precisión en el PC se encuentran descritos en la figura 4.3.

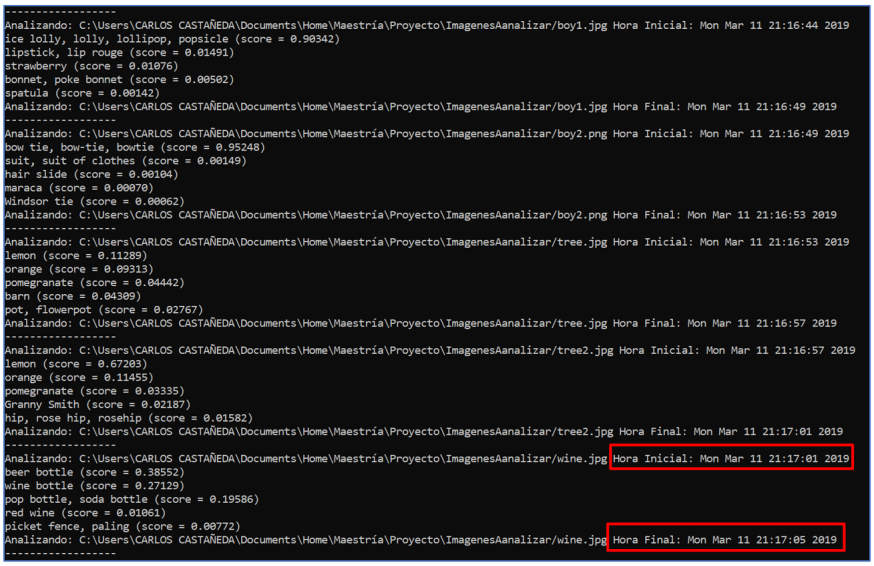


Figura 4.3: Resultados del análisis del Laptop

VPS Linux (t2.micro)

Las características de recurso de hardware disponibles en la máquina se encuentran descritas en la figura 4.4. Para más información ver la documentación oficial de Amazon Web Services

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes

Figura 4.4: Recursos disponibles VPS t2.micro

Los resultados obtenidos del análisis de los datos en cuanto a rendimiento y precisión en la VPS Linux con SO ubuntu se encuentran descritos en la figura 4.5.

VPS Linux (t2.large)

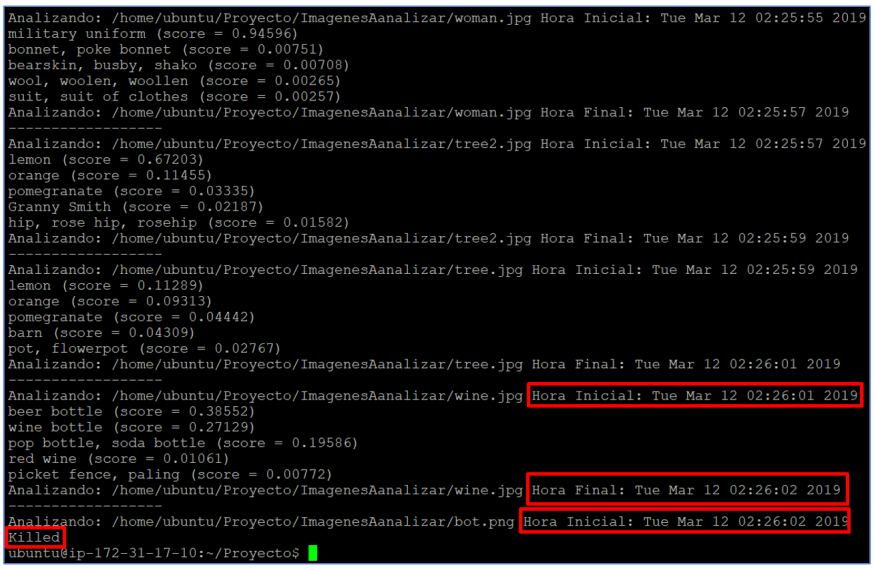


Figura 4.5: Resultados del análisis VPS t2.micro

Las características de recurso de hardware disponibles en la máquina se encuentran descritas en la figura 4.6. Para más información ver la documentación oficial de Amazon Web Services

Instancia	CPU virtual*	Créditos por hora de CPU	Memoria (GiB)	Almacenamiento	Rendimiento de red
t2.nano	1	3	0,5	Solo EBS	Bajo
t2.micro	1	6	1	Solo EBS	De bajo a moderado
t2.small	1	12	2	Solo EBS	De bajo a moderado
t2.medium	2	24	4	Solo EBS	De bajo a moderado
t2.large	2	36	8	Solo EBS	De bajo a moderado
t2.xlarge	4	54	16	Solo EBS	Moderado
t2.2xlarge	8	81	32	Solo EBS	Moderado

Figura 4.6: Recursos disponibles VPS t2.large

Los resultados obtenidos del análisis de los datos en cuanto a rendimiento y precisión en la VPS Linux con SO ubuntu se encuentran descritos en la figura 4.7.

Raspberry

Las características de recurso de hardware disponibles en la máquina se encuentran descritas en la figura 4.8.

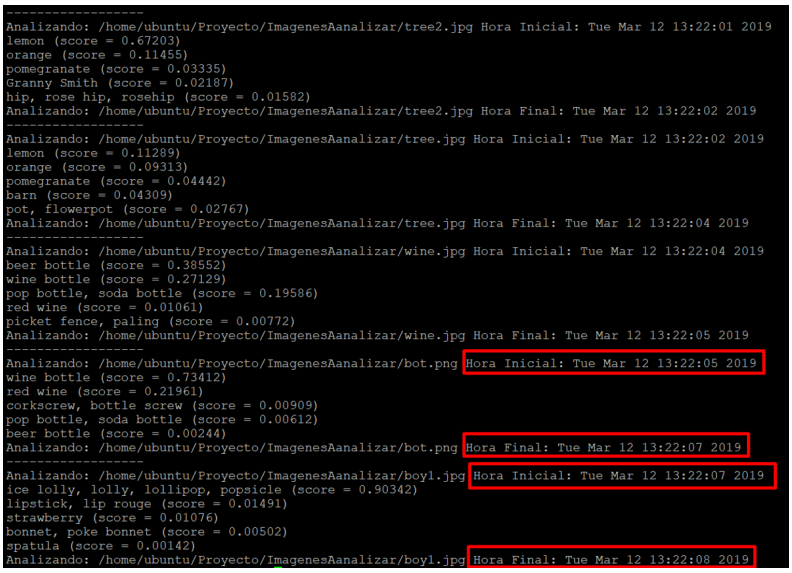


Figura 4.7: Resultados del análisis VPS t2.large

Los resultados obtenidos del análisis de los datos en cuanto a rendimiento y precisión en la Raspberry se encuentran descritos en la figura 4.9.

El resumen de los resultados obtenidos se puede encontrar en la tabla 4.1.

Cuadro 4.1: Resumen de datos del análisis

Dispositivo	RAM	CPU	Tipo de Disco Duro	Precisión	Rendimiento
Laptop	8 GB	i7-8550U	SSD	40 %-70 %	4 Segundos
VPS T2.Micro	1 GB	vCPU 1	SSD	40 %-70 %	2 Segundos
VPS T2.Large	8 GB	vCPU 2	SSD	40 %-70 %	1 Segundos
Raspberry	1 GB	Broadcom BCM2387	Micro SD	40 %-70 %	5 Segundos

De este modelo probado en todos los entornos, bajo las mismas versiones de Python se puede concluir que la cantidad de recursos influye directamente en el rendimiento de ejecución más no tienen relevancia alguna sobre la precisión del modelo ya que en todos los entornos se obtuvieron exactamente los mismos resultados analizando las mismas imágenes.

Al momento de agotar los recursos de máquina el script detiene su ejecución y deja de clasificar las imágenes, este caso se dio en el VPS Ubuntu T2.Micro y en la Raspberry. Como mejoras de rendimiento al script inicialmente utilizado se deja la subida del core de tensorflow en una sola ocasión al principio de la

- Microprocesador:
  - Chipset Broadcom BCM2387.
  - 1,2 GHz de cuatro núcleos ARM Cortex-A53
- Unidad de procesamiento gráfico (GPU: Graphics Processor Unit):
  - Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.
  - Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA
- RAM:
  - 1GB LPDDR2.

Figura 4.8: Recursos disponibles Raspberry

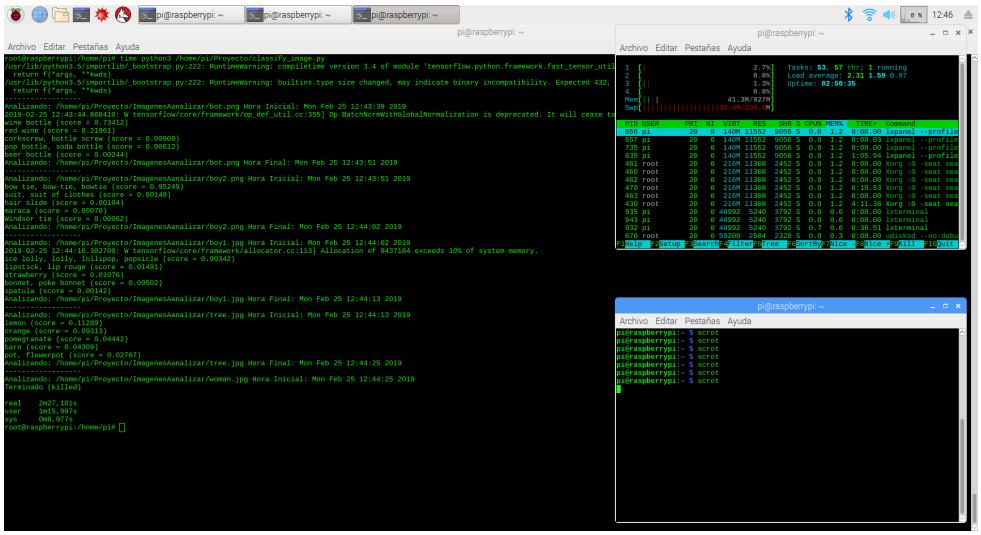


Figura 4.9: Resultados del análisis Raspberry

ejecución y en adelante se realiza una lectura periódica a una carpeta del sistema operativo donde se suben las imágenes a ser analizadas. Además luego de cada análisis de imagen se utiliza la instrucción tf.reset_default_graph() con la finalidad de liberar memoria y dejar los recursos computacionales preparados para ser usados en la siguiente iteración.

Con los resultados encontrados se determina que es posible el uso de Raspberry para realizar el análisis y clasificación de las imágenes, sin embargo los tiempos de respuesta de clasificación de cada una sigue siendo alta, por lo cual se crea un dataset sesgado a alimentos normalmente utilizados en desayunos de cocina Colombiana.

4.2. Fog Computing con Movidious

Luego de analizar los resultados obtenidos se busca la manera de optimizar el rendimiento de la ejecución de los *scripts* en la *Raspberry PI*, se encuentra un *SoC (System on a Chip)* diseñado específicamente para aplicaciones de redes neuronales y de visión por computadora en el dispositivo. La *VPU (Vision Processing Units)* tiene una arquitectura dedicada para el procesamiento de imágenes de alta calidad, la visión por computadora y las redes neuronales profundas, lo que las hace adecuadas para impulsar la combinación exigente de tareas centradas en la visión en los dispositivos inteligentes modernos. Este dispositivo es posible conectarlo vía USB y luego de configuraciones técnicas, instalar el driver y adaptar el modelo de *Tensor Flow* a la *Movidius* se logró correr ejemplos en donde se demuestra que el rendimiento de la red neuronal adaptada al uso del nuevo dispositivo aumenta su rendimiento y hace el mismo trabajo al menos 15 veces más rápido. Para realizar el ejemplo de rendimiento se tomó el ejercicio del reconocimiento de imágenes de los dígitos escritos a mano del MNIST. En la tabla 4.2 se puede encontrar un resumen de los resultados encontrados comparando el rendimiento, medido en segundos, entre la predicción de la *Raspberry* con una *CNN* en *tensorflow* usando únicamente los recursos de máquina y la misma red neuronal adaptada a *Movidius* usando los recursos del *SoC* conectado por *USB* a la *Raspberry*.

Cuadro 4.2: Tiempo de análisis usando movidius

Dígito a inferir	Raspberry (Seg)	Movidius (Seg)
3	0,132107	0,014539
5	0,135376	0,008369
9	0,13192	0,008362
8	0,137825	0,002415
Promedio	0,134307	0,00842125
Relación porcentual	100 %	6,27014973 %
Relación	1	15,9485

Con dicha relación se concluye que el SoC Movidius mejora el rendimiento de la raspberry PI y logra que la categorización de imágenes se realice hasta 15 veces más rápido, en la figura 4.10 se muestra la comparación del tiempo que tardan las predicciones usando únicamente los recursos de la Raspberry y usando los recursos de la Raspberry junto a la Movidious.

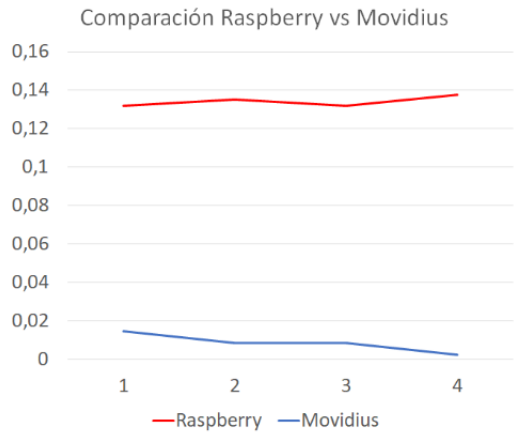


Figura 4.10: Comparación de tiempo usado para analizar una imagen usando los recursos de la Raspberry y sumando los recursos de la Movidious

4.3. Resultados de transmisión y almacenamiento en la nube

El servicio de *AWS IoT Core* opera de manera adecuada para el funcionamiento requerido en la arquitectura, en la imagen 4.11 se comprueba la transmisión realizada desde el dispositivo Raspberry PI actuando como *publisher MQTT* usando un *script* de *python* explicado de manera detallada en la sección 3.3.1, además se simula un suscriptor *MQTT* directamente en la plataforma web de *Amazon Web Services* que permite la visualización de la información recibida como se puede apreciar en la figura 4.12.

```
$ python test.py
Iniciando Lecura de la configuración
Configuración cargada correctamente
Publicado al topic: from_rb_to_server
Mensaje: {"result": "2", "from": "Raspberry", "date": "30/10/2019 09:10:16", "sequence": "swjfclubgw"}
```

Figura 4.11: Ejecución de *Script* de transmisión de datos desde la *Raspberry* hacia la nube usando el protocolo MQTT para publicar en el servicio *AWS IoT Core*

De la misma manera en que se tiene el suscriptor web en la consola de Amazon Web Services, se realiza la configuración del servidor VPS con sistema operativo ubuntu para que se suscriba a todos los mensajes que llegan al servicio de AWS IoT Core a través de un script escrito en Python descrito en la sección 3.3.2. En la figura 4.13 se puede apreciar la recepción del mensaje.

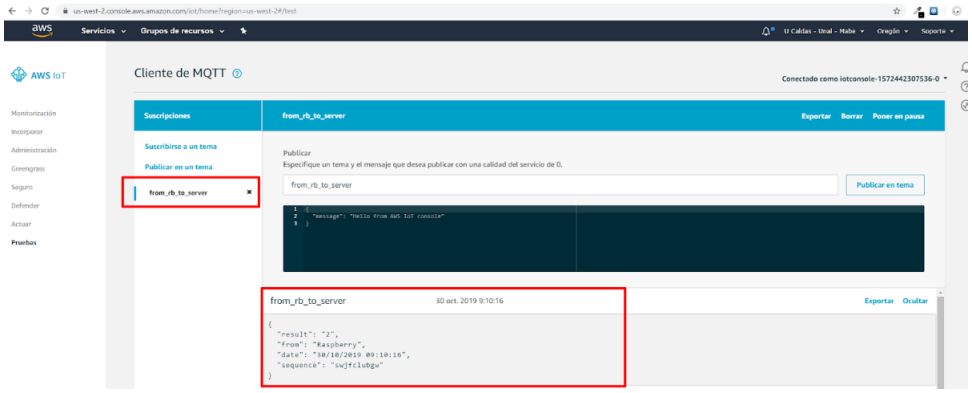


Figura 4.12: Consola del servicio *AWS IoT Core* actuando como suscriptor *MQTT* recibiendo el mensaje de la *Raspberry*

```
ubuntu@ip-172-31-17-10:~/aws-tf-iot-analysis-local/server-side$ python3.7 subToAWSCC.py
Iniciando Lecura de la configuracion
Configuracion cargada correctamente
Suscribiendo a from_rb_to_server
Suscrito a from_rb_to_server


Received a new message:
b'{\r\n  "result": "2",\r\n  "from": "Raspberry",\r\n  "date": "30/11/2019 22:44:47",\r\n  "sequence": "dktebctdlv"\r\n}'
Iniciando Lecura de la configuracion
from topic:
from_rb_to_server
```

Figura 4.13: Consola de VPS EC2 de AWS suscrito al servicio MQTT AWS IoT Core recibiendo un mensaje enviado desde la Raspberry

Luego de recibida la información, se procede a almacenarla en el servicio de MongoDB Atlas, con las configuraciones descritas en la sección 3.3.2, en la figura 4.14 se muestra cómo, a través del cliente de base de datos MongoDB Compass, se visualiza la información ya almacenada con su respectiva fecha y hora de transmisión.

4.4. Resultados prueba de validación con datos de cocina

Esta arquitectura al ser resultado de un macroproyecto fue probada por estudiantes que finalizan publicando un artículo en EVENTO- REVISTA haciendo uso de la arquitectura propuesta en el presente trabajo con un caso de estudio específico en el que se pretende reconocer imágenes tomadas desde la cámara de la raspberry PI a alimentos que se encuentran ubicados en el mise and place, a continuación subir a la nube el dato numérico que indica la categoría de alimento encontrado. Este proceso se realiza con cada alimento que se desea

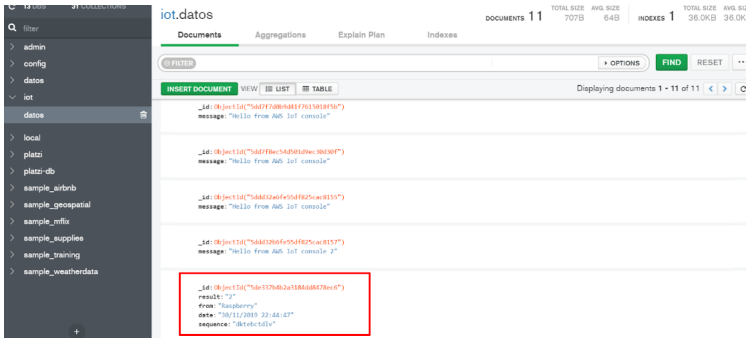


Figura 4.14: Datos almacenados en el motor de bases de datos no relacional MongoDB

agregar a la preparación final. Cuando ya se tienen todos los datos subidos en la nube, de acuerdo a los alimentos recibidos, se clasifica la preparación. Para este caso de estudio los alimentos y los tipos de preparación disponibles se definieron de manera previa.

En la tabla 4.3 se muestran las categorías generales que fueron elegidas para la clasificación de los alimentos y de las preparaciones.

Las pruebas iniciales de validación consistieron en realizar el entrenamiento de una red neuronal convolucional (CNN) para la clasificación de la imagen de los ingredientes, la clasificación de estos se realizó en la *Raspberry PI* haciendo uso de uno de los paradigmas del presente trabajo como lo es el *Fog Computing*. La idea de realizar esta prueba era especificar primero que todo que el modelo no perdiera precisión en la clasificación, que la *Raspberry PI* hiciese uso completo de sus recursos disponibles y que estos a su vez soportan el uso de *tensorflow* paralelizando las tareas de análisis de datos.Finalmente al realizar la clasificación de la imagen se envía el dato numérico del ingrediente clasificado a la nube de *Amazon web services* a través del protocolo elegido de *MQTT*.

En la figura 4.15 se puede apreciar el módulo de visualización creado por los estudiantes de pregrado, en el cual se puede observar los datos recibidos con la clasificación de cada alimento (Parte derecha de la imagen). Al tener una serie de alimentos se procede a realizar la clasificación de la preparación en su conjunto, resultado que es mostrado en la parte superior de la imagen.

Inicialmente la construcción del aplicativo web se realizó bajo el modelo *HTTP* convencional reconocido actualmente en donde se realizaba la subida de imágenes a través de archivos con extensión jpg y posteriormente en el mismo servidor se procedía a analizar cada ingrediente y finalmente la preparación. Observando el funcionamiento del aplicativo se propone cambiar su arquitectura para que se tomen imágenes en tiempo real haciendo uso de la *Raspberry PI*

Figura 4.15: Módulo de visualización de las predicciones de ingredientes y preparaciones

con su módulo de cámara, y en el mismo dispositivo la imagen del alimento sea clasificada aprovechando recursos computacionales del dispositivo IoT restando uso y costos del alojamiento del aplicativo completo en la nube. Al tener la clasificación del alimento se envía el dato de la categoría obtenida (Dato numérico) a la nube de *Amazon Web Services* haciendo uso del protocolo *MQTT* publicando el resultado al servicio *AWS IoT Core*. El VPS con sistema operativo *linux* se encuentra suscrito al tema de publicación MQTT de la raspberry y recibe el dato desde *AWS IoT Core*, almacenando en base de datos y colocándolo en cola de los alimentos que se están alistando en la preparación, estos alimentos se pueden ver en el módulo de visualización web mostrado anteriormente. Cuando el usuario considere que ya tiene la totalidad de los ingredientes selecciona la opción de predecir la preparación y es allí donde el servidor *VPS* a través de los modelos previamente entrenados predice la preparación que se está realizando, para este caso específico el tipo de desayuno.

4.5. Resultados del modelo de redes neuronales convolucionales para el análisis de imágenes

Para el análisis de imágenes en el dispositivo *Raspberry PI* y la aplicación de la técnica de *machine learning* llamada *deep learning* se usó una red neuronal convolucional. Posterior al entrenamiento de la red neuronal es posible visualizar la precisión del modelo que se presenta en la figura 4.16 y la pérdida que se presenta en la figura 4.17

Finalmente en la figura 4.18 se puede visualizar las curvas ROC que permite evaluar el rendimiento del modelo. La curva ROC muestra la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) del modelo. TPR describe la tasa a la que el clasificador predice como “positivo” a las observaciones que son “positivas”. FPR describe la tasa a la que el clasificador predice como “positivo” a las observaciones que en realidad son “negativas”. Los clasificadores perfectos tienen una TPR de 1 y una FPR de 0.

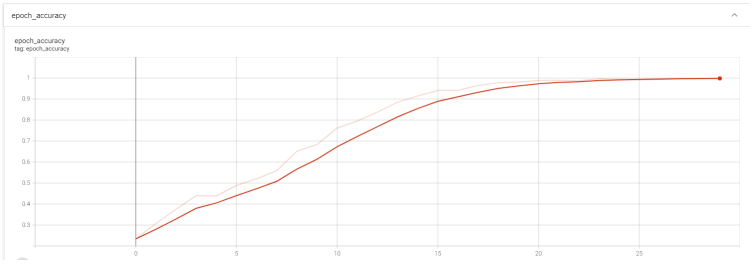


Figura 4.16: Epoch accuracy

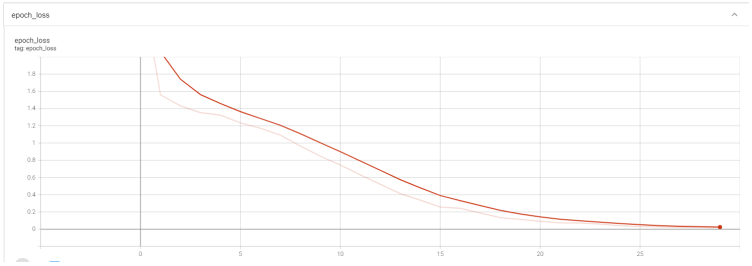


Figura 4.17: Epoch loss

4.6. Resumen

Luego realizar la parte experimental de la arquitectura se concluye que con los servicios y dispositivos elegidos (Ver tablas 3.5, 3.6) se logró cubrir las tres áreas definidas inicialmente como reto de la arquitectura. Es importante dejar claro que en la parte experimental se eligió el proveedor de nube *Amazon Web Services* pero como se definió en la sección 3.1.1 se pueden usar otros proveedores con otros servicios propuestos.

Al interno del hogar se logró transmitir datos a través del protocolo *MQTT* en donde el dispositivo *Raspberry PI* actuaba como *Broker MQTT* recibiendo los datos de todos los dispositivos *IoT* de la cocina. Al momento de transmitir desde el hogar a la nube, se usó también *MQTT* pero esta vez con una capa adicional de seguridad bajo el protocolo TLS/SSL, la transmisión se realizó directamente al servicio *AWS IoT Core* en donde la *Raspberry PI* publicaba los datos procesados y en la nube se recibían por parte del suscriptor que para este caso era un *VPS* con sistema operativo *Ubuntu* que posteriormente se encargaba de almacenar los datos en el servicio de base de datos no relacional *MongoDB Atlas*.

Se logró analizar datos en el dispositivo *Raspberry PI* usando *tensorflow* y aprovechando la totalidad de los recursos computacionales del dispositivo (Ver imagen 4.1) aplicando el concepto de *fog computing*; el resultado del análisis estaba acorde al entrenamiento de la red neuronal y los resultados obtenidos

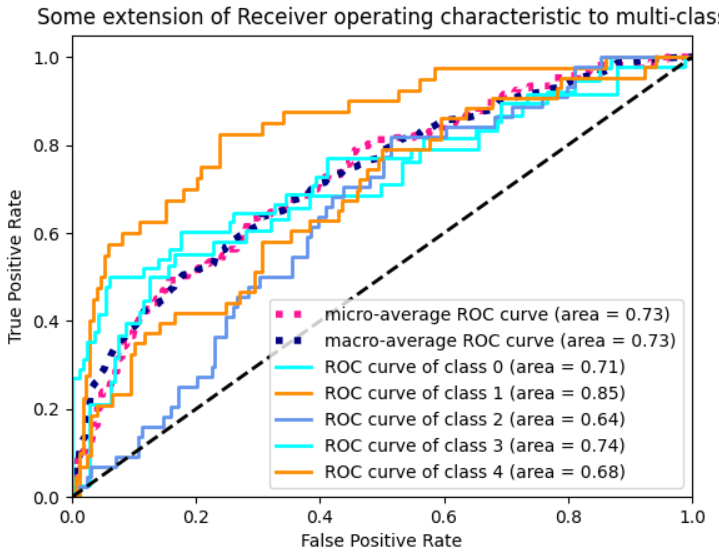


Figura 4.18: Curvas ROC

continuaban con la misma precisión inicial del modelo. En este dispositivo se usaron redes neuronales convolucionales para el análisis de imágenes capturadas directamente desde la cámara con la finalidad de clasificarlas y enviar el resultado de la clasificación a la nube evitando la transmisión completa de la imagen. Al tener la información almacenada en la base de datos no relacional, el *VPS* tenía instalada una aplicación web desde la cual se podía accionar el análisis de los datos para categorizar la preparación realizar basado en los ingredientes previamente clasificados en la *Raspberry PI*, este análisis se realiza usando *SVM* sobre datos en texto plano.

Finalmente se logró almacenar la información en el motor de bases de datos no relacionales *Mongo DB* en el servicio de *MongoDB Atlas*, el almacenamiento y posterior consulta de los datos se realizaba directamente desde el *VPS*.

Cuadro 4.3: Categorías de las predicciones

Categoría	Elementos
Categoría A: Ingredientes	▪ Huevos ▪ Arepas ▪ Mantequilla ▪ Chocolate ▪ Pan ▪ Cereales ▪ Café ▪ Leche ▪ Tocino ▪ Changua ▪ Tamal ▪ Papas ▪ Calentado ▪ Yuca Frita ▪ Jugo naranja ▪ Yogurt ▪ Pollo
Categoría B: Recetas de desayunos	▪ Paisa ▪ Cafetero ▪ Rolo ▪ Fitness ▪ Tolimense ▪ Americano

Capítulo 5

Conclusiones y trabajo futuro

- El análisis de datos tiene una serie de algoritmos y parámetros que han sido probados en diferentes contextos, los resultados de los mismos y las pruebas realizadas en este trabajo confirman que, por ejemplo, las redes neuronales convolucionales tienen mayor rendimiento frente a las demás técnicas cuando la fuente de los datos son imágenes. Sobre este punto es importante referirse a la tabla 3.7 en donde se muestra, basado en literatura, cual es el algoritmo recomendado de acuerdo al tipo de dato a ser analizado.
- Para el caso del hogar conectado refiriéndose específicamente al caso de estudio de este trabajo, en el cual en la cocina se realizaba, entre otros, captura de imágenes se concluye que los dispositivos *IoT* como la *Raspberry PI* soportan el análisis de datos usando técnicas de descubrimiento de información como las redes neuronales convolucionales, sin embargo, es importante resaltar que el rendimiento de la misma frente otros dispositivos con mayor capacidad computacional no es igual y en algunas pruebas los dispositivos se ven sin capacidad y se reinician en medio del análisis. Cabe aclarar que los resultados del análisis conservan la precisión dado que los modelos son entrenados externamente y posteriormente son pasados al dispositivo.
- Se hicieron pruebas de análisis sobre fuentes de datos de tipo texto que gracias a sus características requieren menor cantidad de recursos computacionales y el dispositivo *Raspberry PI* respondió de manera correcta a las solicitudes. En este punto es importante aclarar que el *dataset* usado se encontraba sesgado al problema específico de cocina que se estaba resolviendo en este trabajo y la cantidad de datos analizados no era de gran volumen. Como trabajo futuro quedará probar con un *dataset* que tenga mayor volumen de datos y validar si el rendimiento del dispositivo se mantiene.
- La arquitectura propuesta tiene como características principales, la trans-

misión segura de datos, a través del protocolo *MQTT*, el almacenamiento de datos no estructurado usando servicios como *MongoDB Atlas*, y el procesamiento de los datos usando un servidor *VPS*, estos servicios son soportados por las principales plataformas computacionales que permiten el procesamiento en la nube (Amazon Web Services, Microsoft Azure y Google Cloud Platform) como se puede visualizar en el *benchmarking* plasmado en las tablas 3.2, 3.3 y 3.1. Todos los proveedores nube consultados poseen servicios para la transmisión segura de datos desde dispositivos IoT a través del protocolo *MQTT*, permiten almacenar datos no estructurados bajo bases de datos convencionales o propias de cada nube y contienen servicio de VPS para suscribirse al tema de *MQTT*, recibir y analizar la información. La arquitectura se trabajó desde la generalización y se probó en *Amazon Web Services* concluyendo que la concepción de la misma se presta para trabajar en los diferentes proveedores nube con sus servicios específicos.

- En este trabajo se pudo comprobar que hay compatibilidad entre las diferentes implementaciones de *Tensorflow* en la misma versión tanto dispositivos IoT como en servidores, los resultados son los mismos, lo que varia es el tiempo de ejecución por la diferencia de capacidad de procesamiento de los dispositivos.

- La arquitectura propuesta soporta implementación en diferentes campos que tengan dispositivos *IoT* sensando, analizando, transmitiendo y almacenando información, se propone como trabajo futuro probar la arquitectura en diferentes contextos y con diferentes técnicas de análisis de datos.

- La recepción de la información desde el *Broker MQTT* actualmente se realiza en el *VPS*, sin embargo esta podría ser realizada directamente en servicios *serverless* lo que permitiría desacoplar la transmisión de los datos del *VPS* enfocándolo en el análisis de los datos. Se propone que para trabajos futuros la arquitectura sea probando usando servicios *serverless*, para el caso del proveedor de nube *AWS* podría usarse servicios *lambda*.

- Como trabajo futuro se propone probar el análisis de datos de la arquitectura (Tanto en el *fog computing* como en el *VPS*) con otros *frameworks* de analítica como *pytorch*. Adicionalmente se propone probar el análisis de datos en servicios *serverless* entendiendo las limitaciones de recursos y tiempos de ejecución con las que actualmente cuentan las mismas. De lograr esto los servicios quedarían totalmente desacoplados y la arquitectura del lado de la nube podría ser actualizada.

Bibliografía

[Alam et al., 2016] Alam, F., Mehmood, R., Katib, I., and Albeshri, A. (2016). Analysis of eight data mining algorithms for smarter internet of things (iot). *Procedia Computer Science*, 98:437–442.

[Albahri et al., 2021] Albahri, A., Alwan, J. K., Taha, Z. K., Ismail, S. F., Hamid, R. A., Zaidan, A., Albahri, O., Zaidan, B., Alamoodi, A., and Alsalem, M. (2021). Iot-based telemedicine for disease prevention and health promotion: State-of-the-art. *Journal of Network and Computer Applications*, 173:102873.

[Alsheikh et al., 2014] Alsheikh, M. A., Lin, S., Niyato, D., and Tan, H.-P. (2014). Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018.

[Bandyopadhyay and Bhattacharyya, 2013] Bandyopadhyay, S. and Bhattacharyya, A. (2013). Lightweight internet protocols for web enablement of sensors using constrained gateway devices. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 334–340. IEEE.

[Basu et al., 2002] Basu, S., Banerjee, A., and Mooney, R. (2002). Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer.

[Betancourt, 2005] Betancourt, G. A. (2005). Las máquinas de soporte vectorial (svms). *Scientia et technica*, 1(27).

[Bifet et al., 2010] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604.

[Bin et al., 2010] Bin, S., Yuan, L., and Xiaoyi, W. (2010). Research on data mining models for the internet of things. In *2010 International Conference on Image Analysis and Signal Processing*, pages 127–132. IEEE.

[Bkassiny et al., 2012] Bkassiny, M., Li, Y., and Jayaweera, S. K. (2012). A survey on machine-learning techniques in cognitive radios. *IEEE Communications Surveys & Tutorials*, 15(3):1136–1159.

[Bonomi et al., 2012] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

[Botta et al., 2016] Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700.

[Buczak and Guven, 2015] Buczak, A. L. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176.

[Carbonell et al., 1983] Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). An overview of machine learning. In *Machine learning*, pages 3–23. Elsevier.

[Castañeda et al., 2019] Castañeda, D. S. et al. (2019). *Aplicación de support vector machine al mercado colombiano*. PhD thesis, Universidad del Rosario.

[Chen et al., 2014a] Chen, M., Mao, S., and Liu, Y. (2014a). Big data: A survey. *Mobile networks and applications*, 19(2):171–209.

[Chen et al., 2014b] Chen, S., Xu, H., Liu, D., Hu, B., and Wang, H. (2014b). A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4):349–359.

[Di et al., 2019] Di, Z., Gong, X., Shi, J., Ahmed, H. O., and Nandi, A. K. (2019). Internet addiction disorder detection of chinese college students using several personality questionnaire data and support vector machine. *Addictive Behaviors Reports*, 10:100200.

[Ding et al., 2013] Ding, G., Wang, L., and Wu, Q. (2013). Big data analytics in future internet of things. *arXiv preprint arXiv:1311.4112*.

[Dos Santos and Gatti, 2014] Dos Santos, C. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

[Espinosa-Oviedo et al., 2017] Espinosa-Oviedo, J. E., Zuluaga-Mazo, A., and Gómez-Montoya, R. A. (2017). Kernel methods for improving text search engines transductive inference by using support vector machines. *Tecciencia*, 12(22):51–60.

[Fatima et al., 2017] Fatima, M., Pasha, M., et al. (2017). Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications*, 9(01):1.

[Foody, 2004] Foody, G. M. (2004). Thematic map comparison. *Photogrammetric Engineering & Remote Sensing*, 70(5):627–633.

[Foody and Mathur, 2006] Foody, G. M. and Mathur, A. (2006). The use of small training sets containing mixed pixels for accurate hard image classification: Training on mixed spectral responses for classification by a svm. *Remote Sensing of Environment*, 103(2):179–189.

[Graves et al.,] Graves, A., Mohamed, A., and Hinton, G. Speech recognition with deep recurrent neural networks. in 2013 ieee int. In *Conf. on Acoustics, Speech And Signal Processing*, pages 6645–6649.

[Guzmán et al., 2017] Guzmán, I. C., Oslinger, J. L., and Darío Nieto, R. (2017). Wavelet denoising of partial discharge signals and their pattern classification using artificial neural networks and support vector machines. *Dyna*, 84(203):240–248.

[Hurtado et al., 2002] Hurtado, J., Henao, R., and Castellanos, G. (2002). Clasificación de señales sísmicas por medio de onditas y máquinas de soporte vectorial. *Universidad Nacional de Colombia. Colombia. Observatorio Sismológico de Quindio. Colombia. INGEOMINAS. sfsl CO*.

[Jaramillo Garzón, 2013] Jaramillo Garzón, J. A. (2013). Protein function prediction with semi-supervised classification based on evolutionary multiobjective optimization. *Departamento de Ingeniería Eléctrica, Electrónica y Computación*.

[Jipkate and Gohokar, 2012] Jipkate, B. R. and Gohokar, V. (2012). A comparative analysis of fuzzy c-means clustering and k means clustering algorithms. *International Journal Of Computational Engineering Research*, 2(3):737–739.

[Kalchbrenner et al., 2014] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lane et al., 2015] Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., and Kawsar, F. (2015). An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *Proceedings of the 2015 international workshop on internet of things towards applications*, pages 7–12.

[Lawrence et al., 1997] Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

[Leal et al., 2016] Leal, J. A., Ochoa, L. H., and García, J. A. (2016). Identification of natural fractures using resistive image logs, fractal dimension and support vector machines. *Ingeniería e Investigación*, 36(3):125–132.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Li and Yu, 2011] Li, B. and Yu, J. (2011). Research and application on the smart home based on component technologies and internet of things. *Procedia Engineering*, 15:2087–2092.

[Likas et al., 2003] Likas, A., Vlassis, N., and Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461.

[Marsland, 2015] Marsland, S. (2015). *Machine learning: an algorithmic perspective*. CRC press.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[Moeskops et al., 2016] Moeskops, P., Viergever, M. A., Mendrik, A. M., De Vries, L. S., Benders, M. J., and Išgum, I. (2016). Automatic segmentation of mr brain images with a convolutional neural network. *IEEE transactions on medical imaging*, 35(5):1252–1261.

[Musumeci et al., 2018] Musumeci, F., Rottondi, C., Nag, A., Macaluso, I., Zibar, D., Ruffini, M., and Tornatore, M. (2018). An overview on application of machine learning techniques in optical networks. *IEEE Communications Surveys & Tutorials*, 21(2):1383–1408.

[Naik, 2017] Naik, N. (2017). Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE.

[Nguyen and Armitage, 2008] Nguyen, T. T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4):56–76.

[Pal and Mather, 2005] Pal, M. and Mather, P. (2005). Support vector machines for classification in remote sensing. *International journal of remote sensing*, 26(5):1007–1011.

[Renart et al., 2019] Renart, E. G., Veith, A. D. S., Balouek-Thomert, D., De Assuncao, M. D., Lefevre, L., and Parashar, M. (2019). Distributed operator placement for iot data analytics across edge and cloud resources.

[Sak et al., 2014] Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

[Sehgal and Kehtarnavaz, 2018] Sehgal, A. and Kehtarnavaz, N. (2018). A convolutional neural network smartphone app for real-time voice activity detection. *IEEE Access*, 6:9017–9026.

[Severyn and Moschitti, 2015] Severyn, A. and Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962.

[Shi et al., 2017] Shi, Z., Shi, M., and Li, C. (2017). The prediction of character based on recurrent neural network language model. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 613–616. IEEE.

[Stergiou et al., 2018] Stergiou, C., Psannis, K. E., Kim, B.-G., and Gupta, B. (2018). Secure integration of iot and cloud computing. *Future Generation Computer Systems*, 78:964–975.

[Stolpe, 2016] Stolpe, M. (2016). The internet of things: Opportunities and challenges for distributed data analysis. *ACM SIGKDD Explorations Newsletter*, 18(1):15–34.

[Szepesvári, 2010] Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

[Wagstaff et al., 2001] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584.

[Wilches-Cortina et al., 2017] Wilches-Cortina, J. R., Cardona-Peña, J. A., and Tello-Portillo, J. P. (2017). A voip call classifier for carrier grade based on support vector machines. *Dyna*, 84(202):75–83.

[Wu et al., 2014] Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., and Long, K. (2014). Cognitive internet of things: a new paradigm beyond connection. *IEEE Internet of Things Journal*, 1(2):129–143.

[Yuan et al., 2019] Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824.

[Zhang et al., 2019] Zhang, Y.-D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K., and Wang, S.-H. (2019). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimedia Tools and Applications*, 78(3):3613–3632.

[Zhong et al., 2021] Zhong, M., Zhou, Y., and Chen, G. (2021). Sequential model based intrusion detection system for iot servers using deep learning methods. *Sensors*, 21(4).

[Zhu and Goldberg, 2009] Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.