Aprendizaje profundo para datos multimedia en IoT

Srinidhi Hiriyannaiah, BS Akanksh, AS Koushik, GM Siddesh y KG Srinivasa

Resumen Con la llegada de Internet que ha llevado a la proliferación de grandes cantidades de datos multimedia, el análisis de los datos multimedia agregados ha demostrado ser una de las áreas activas de investigación y estudio. Los datos multimedia incluyen audio, video, imágenes asociadas con aplicaciones como búsquedas de similitudes, resolución de entidades y clasi fi cación. La minería de datos visual es ahora uno de los campos de aprendizaje activo que incluye aplicaciones de vigilancia para detección de objetos, detección de fraude, detección de delitos y otras aplicaciones. La minería de datos multimedia incluye muchos desafíos como el volumen de datos, la variedad y la naturaleza no estructurada, no estacionaria y en tiempo real. Necesita capacidades de procesamiento avanzadas para tomar decisiones casi en tiempo real. Los sistemas de bases de datos tradicionales existentes, las técnicas de minería de datos no se pueden utilizar debido a sus limitaciones. Por eso, Para procesar cantidades tan grandes de datos, se pueden utilizar técnicas avanzadas como el aprendizaje automático, métodos de aprendizaje profundo. Los datos multimedia también incluyen datos de sensores que se generan ampliamente. La mayoría de las aplicaciones sanitarias incluyen sensores para detectar la frecuencia cardíaca, la presión arterial y la frecuencia del pulso. El avance de los teléfonos inteligentes ha dado como resultado aplicaciones basadas en el fitness basadas en el número de pasos caminados, el recuento de calorías, los kilómetros recorridos, etc. Todos estos tipos de datos pueden clasificarse como datos multimedia para Internet de las cosas (IoT). Hay muchos dispositivos de interfaz que están interconectados entre sí con la red troncal como una red informática cuando se trata de datos de sensores. El objetivo principal de este capítulo es destacar la importancia y la convergencia de las técnicas de aprendizaje profundo con IoT. Se pone énfasis en la clasificación Los datos multimedia también incluyen datos de sensores que se generan ampliamente. La mayoría de las aplicaciones sanitarias incluyen sensores para detectar la frecuencia cardíaca, la presión arterial y la frecuencia del pulso. El avance de los teléfonos inteligentes ha dado como resultado aplicaciones basadas en el fitness basadas en el número de pasos caminados, el recuento de calorías, los kilómetros recorridos, etc. Todos estos tipos de datos pueden clasificarse como datos multimedia para Internet de las cosas (IoT). Hay muchos dispositivos de interfaz que están interconectados entre sí con la red troncal como una red informática cuando se trata de datos de sensores. El objetivo principal de este capítul

S. Hiriyannaiah (B) · BS Akanksh · COMO Koushik · Instituto de Tecnología GM Siddesh Ramaiah, Bangalore, India
Email: srinidhi.hiriyannaiah@gmail.com

BS Akanksh
Email: bsakanksh@gmail.com

COMO Koushik
Email: askoushik4@gmail.com

GM Siddesh
Email: siddeshgm@gmail.com

KG Srinivasa
Instituto Nacional de Investigación Técnica de Formación Docente, Nueva Delhi, India correo electrónico: kgsrinivasa@gmail.com

S. Tanwar y col. (eds.), *Computación de Big Data multimedia para aplicaciones de IoT*, Biblioteca de referencia de sistemas inteligentes 163,
https://doi.org/10.1007/978-981-13-8759-3_4

de datos de IoT mediante el aprendizaje profundo y el ajuste fino esencial de los parámetros. Se utiliza un dispositivo sensor virtual implementado en Python para la simulación. Se analiza brevemente una descripción de los protocolos utilizados para la comunicación de dispositivos IoT. Se proporciona un estudio de caso sobre la clasificación del conjunto de datos de calidad del aire mediante técnicas de aprendizaje profundo.

Palabras clave Datos multimedia · IoT · Análisis de la calidad del aire · Aprendizaje profundo · Análisis de IoT

1 Introducción a multimedia e IoT

La minería de datos ha ganado una mayor importancia en los últimos días debido a la gran cantidad de datos que se recopilan y su disponibilidad a través de Internet. Los importantes avances en la tecnología y las herramientas de big data han llevado al desarrollo de análisis e investigación en esta área a lo largo de los años. Hay un cambio de paradigma de la minería de datos hacia el análisis de big data que involucra varias etapas de procesamiento y análisis. La analítica de big data consiste en la exploración de datos, identificando las relaciones entre las diferentes características de los datos y visualizándolas. Las aplicaciones de la analítica de big data incluyen varios campos multidisciplinarios como el aprendizaje automático, la recuperación de información, las bases de datos y la visualización. Estos importantes avances han llevado a la evolución de los sistemas de gestión multimedia.

Los datos multimedia incluyen imágenes, video, audio y texto. Debido a la llegada de los teléfonos inteligentes e Internet, este tipo de datos multimedia debe conducir a varias aplicaciones y plataformas para compartir [1]. Las fuentes valiosas de datos multimedia son las redes sociales como Instagram, YouTube, Twitter y Facebook [2]. El volumen de datos que se recopilan en estos sitios aumenta significativamente día a día. Por ejemplo, en Instagram los usuarios han subido más de 20 mil millones de fotos, 100 videos se cargan cada minuto por día en YouTube, 500 millones de tweets en Twitter. Esta enorme cantidad de información es una fuente importante para analizar diferentes patrones y desarrollar aplicaciones. Esta rica fuente de información se denomina "Big data". Los macrodatos poseen tres características esenciales: variedad, velocidad y volumen. La proliferación de big data en términos de volumen, velocidad y variedad ha alcanzado diferentes dominios, como se muestra en la Fig. 1 . En este capítulo, la atención se centra en las aplicaciones de IoT y multimedia.

Internet de las cosas (IoT) es una red de cosas que están conectadas entre sí y que es compatible con Internet para la comunicación. La mayoría de los dispositivos de hoy en día están basados en sensores y normalmente están conectados a uno o más. En la mayoría de las revisiones se estima que el total de dispositivos que estarán conectados entre sí será de 20 mil millones para 2020 [3]. Con el aumento en la cantidad de dispositivos que están conectados entre sí, también aumenta la generación de datos multimedia. El alcance de IoT no se limita solo a los datos del sensor, sino que también se relaciona con multimedia. Por ejemplo, la cámara CCTV captura los datos de video con fines de vigilancia que se pueden clasificar como datos multimedia. Los datos de huellas dactilares también se pueden considerar como datos multimedia ya que están en forma de imagen. Las aplicaciones de dichos datos se utilizan en sistemas inteligentes.

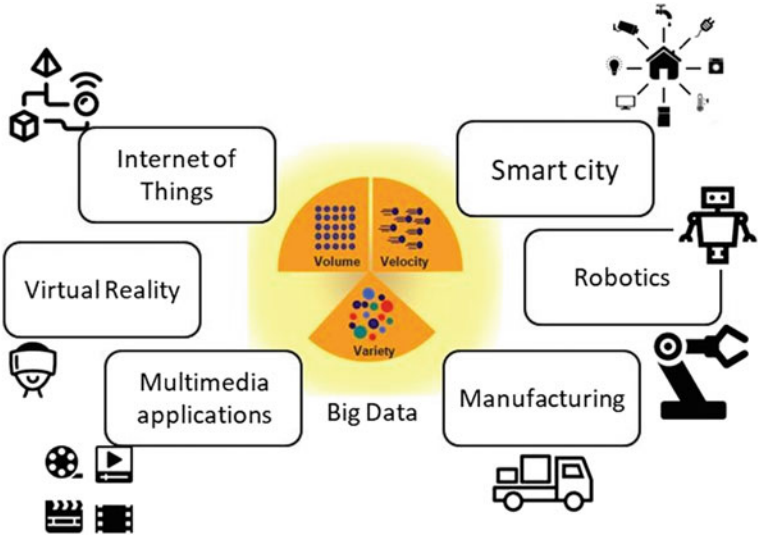


Figura 1 Big data y aplicaciones

comunidades como ciudad inteligente, energía inteligente, red inteligente, etc. [3]. Por lo tanto, la analítica multimedia también juega un papel crucial en el campo de IoT.

La analítica multimedia se puede llevar a cabo utilizando enfoques analíticos convencionales, como agrupamiento, métodos de árbol de decisiones. Sin embargo, un desafío importante al que se enfrentan los enfoques convencionales es la escalabilidad de los métodos y el tiempo de ejecución necesario para producir los resultados [4]. En este sentido, los métodos de computación GPU y Deep Learning se utilizan para análisis multimedia. Los métodos de aprendizaje profundo que involucran redes de avance profundo, las redes recurrentes se utilizan la mayoría de las veces para análisis multimedia. Las características que están presentes en los datos multimedia deben extraerse para poder tomar una decisión clara. El principal desafío del aprendizaje profundo es que el proceso de extracción de características involucra etapas de reconocimiento de objetos, detección de objetos desde los bordes identificados en la imagen.

Las aplicaciones multimedia necesitan un tipo de enfoque de análisis diferente en comparación con las técnicas analíticas convencionales [4]. Las técnicas convencionales de análisis de datos, como la agrupación, la regresión y la clasificación de bayes ingenua se pueden utilizar en los datos estacionarios. Dado que los datos multimedia y los datos de IoT no son estacionarios, las técnicas que se basan en el análisis de transmisión son útiles. Los diferentes tipos de tecnologías habilitadoras para el análisis multimedia y la clasificación de datos de IoT se tratan en las secciones siguientes.

El objetivo de este capítulo es presentar los avances en datos multimedia, los desafíos en el análisis multimedia y cómo se pueden utilizar los métodos de aprendizaje profundo para el análisis multimedia. Se proporciona una ilustración sobre la comunicación de dispositivos IoT. Se analiza una breve descripción del protocolo MQTT y sus operaciones fundamentales, como publicar y suscribirse. En la última sección del capítulo, un caso

Se presenta un estudio sobre clasificación de datos de IoT. El conjunto de datos elegido para este propósito es el conjunto de datos de calidad del aire. Se trata de crear un modelo de aprendizaje profundo para clasificar los datos de IoT junto con el código asociado.

2 Avances en datos multimedia

En el mundo de la analítica de datos, los datos se han multiplicado de diferentes formas. Los datos multimedia son uno de esos datos que se han generado recientemente en grandes volúmenes en Internet. Está siendo capturado por varios dispositivos informáticos multimedia como computadoras, tabletas, teléfonos móviles y cámaras [5]. La naturaleza de los datos que se capturan también es ubicua. Inicialmente, comenzó con audio, video ahora ha llegado a animaciones en forma de gifs. Los avances en datos multimedia se resumen a continuación.

- **Datos visuales:** La forma más común de datos multimedia que se encuentran son los datos de video. Un dato visual consiste en una secuencia de secuencias de imágenes que deben analizarse una por una. La mayor parte de la información no estructurada existe en forma de datos visuales y contiene información muy rica. El proceso de análisis de datos visuales implica extraer información significativa de las diferentes secuencias de imágenes que están presentes en los datos visuales. Sin embargo, el principal desafío radica en el tamaño de los datos visuales. Las tecnologías recientes como la computación en la nube, la computación de alto rendimiento han permitido la investigación de datos visuales y análisis en áreas como los sistemas de videovigilancia, los sistemas autónomos y la atención médica. Los avances en análisis y datos visuales están desafiando al cerebro humano y su computación. En [5] una de las competiciones

  sostenida, la máquina superó a los humanos en la clasificación de imágenes.
- **Datos de audio:** Otro tipo de datos multimedia que se utiliza principalmente es el audio / voz.
  datos. Se necesitan aplicaciones analíticas de audio en tiempo real en las redes sociales, la atención médica y la industria. El análisis de audio implica extraer información útil de las diferentes partes de la información presente en los datos de audio. Los centros de llamadas son una de las aplicaciones de la industria que necesitan análisis de audio para interactuar con el cliente y capacitar a las personas involucradas en el centro de llamadas. Las plataformas de big data como Spark, Hadoop y diferentes bibliotecas se utilizan ampliamente para este tipo de análisis de audio.

  aplicaciones.
- **Datos de texto:** Los datos multimedia se pueden incrustar en el contexto textual en la forma
  de páginas web, encuestas, feeds y metadatos. El análisis de dichos datos ayuda a obtener información interesante. Los datos multimedia en forma de texto pueden estar estructurados o no estructurados. El tipo de datos estructurados se puede analizar con la ayuda de las técnicas tradicionales de recuperación de consultas de bases de datos relacionales. Sin embargo, los datos multimedia en forma de feeds no están estructurados y deben transformarse en un formato estructurado para su posterior análisis [6]. Una de las aplicaciones de ejemplo de análisis de texto multimedia se basa en una situación particular como elecciones, desastres naturales, mercado de valores, etc. Las emociones detrás del texto se pueden analizar con la ayuda del análisis de datos multimedia. De esta manera, se pueden extraer diferentes tipos de información de diferentes fuentes de datos textuales multimedia.

- **Datos del sensor:** IoT está desempeñando un papel importante hoy en día y los sensores están presentes en casi todas partes. Los sensores están equipados no solo para capturar los datos, sino también para aplicar análisis en tiempo real [7]. Con los avances en el hardware y las tecnologías de la computación en nube, los datos de los sensores están aumentando enormemente. Es muy difícil analizar estos datos y desarrollar una aplicación analítica basada en ellos. Las aplicaciones de datos de sensores son muy apreciadas en las ciencias astronómicas para patrones meteorológicos, acondicionamiento de satélites y monitoreo. En el sector sanitario, la mayoría de las aplicaciones se basan en datos de sensores. Por lo tanto, con los avances en los datos de los sensores, es muy esencial desarrollar aplicaciones analíticas basadas en

  eso.
- **Redes sociales:** La principal fuente de datos multimedia son las redes sociales. los
  los avances en las redes sociales y el intercambio que comenzaron con un texto normal ahora han llegado a imágenes, videos, videos en vivo, grupos públicos, etc. Aplicaciones de recomendación [7] utilizan ampliamente el contenido multimedia disponible en las redes sociales para brindar recomendaciones mediante el análisis de los mensajes, video, audio y texto compartidos. Los servicios de personalización son más utilizados por los usuarios de teléfonos inteligentes en función de las suscripciones realizadas por ellos.

La principal característica de los datos multimedia es la variedad. Existe en diferentes formas y en diversas fuentes. Estos avances en datos y análisis multimedia han permitido varios desafíos y tecnologías para desarrollar diferentes aplicaciones. Sin embargo, existen varias tecnologías para el análisis multimedia, los desafíos que se presentan para el análisis son mayores y deben abordarse con cuidado. En la siguiente sección, se discuten los diferentes desafíos que existen para el análisis multimedia, seguidos de las tecnologías habilitadoras para el análisis multimedia.

3 desafíos en datos multimedia

Los datos multimedia involucran los datos de diversas fuentes como cámaras, redes sociales, sensores y otras de naturaleza heterogénea. Con el fin de llevar a cabo análisis de dichos datos, la naturaleza de heterogeneidad de dichos datos debe transformarse con fines de análisis. La transformación de los datos implica convertir los datos de diferentes formatos a un formato singular para el análisis. Algunos de los desafíos que generalmente se ven con el análisis multimedia se analizan a continuación.

3.1 Adquisición y volumen de datos

Los datos multimedia requieren una gran cantidad de almacenamiento para análisis e implican la adquisición de datos de varias fuentes. Hay diferentes fuentes heterogéneas que están involucradas, como redes sociales, sensores, dispositivos móviles, cámaras y mundos virtuales [4]. Los datos heterogéneos que se generan a partir de estas fuentes son múltiples

de naturaleza estructural y multimodal. Cada una de las fuentes exhibe diferentes tipos de características y una adquisición altamente compleja tanto en términos de cantidad como de calidad. Se necesitan nuevas técnicas y tecnologías para comprender la naturaleza variable de los datos multimedia para la adquisición [4 , 6].

Con el crecimiento sin precedentes de los datos multimedia, existe un gran desafío para el almacenamiento y su disponibilidad para aplicaciones multimedia. Se han realizado muchas investigaciones para abordar este desafío en los datos multimedia. En [7], se introduce una canalización de procesamiento de big data junto con MapReduce que consta de etapas como el preprocesamiento de datos, el reconocimiento de datos y la reducción de carga. Las bases de datos NoSQL juegan un papel importante en el almacenamiento de datos multimedia que permite la flexibilidad del esquema involucrado en el almacenamiento. Las diferentes tecnologías habilitadoras que pueden manejar este problema se tratan en la siguiente sección.

3.2 Extracción de características

Los datos multimedia, como audio y video, incluyen numerosas funciones. El proceso de extraer las características de estos datos juega un papel importante en el análisis multimedia. Las diferentes características del video pueden ser el color, el borde, la forma y la textura de las secuencias de imágenes del video. La extracción de características se divide en dos categorías, a saber, extracción de características locales y extracción de características globales. Las técnicas de extracción de características globales implican obtener el histograma de color de diferentes objetos que están presentes en la secuencia de imágenes del video. El desarrollo de histogramas de color ayuda a identificar los bordes de los objetos para distinguirlos. Sin embargo, implica una gran cantidad de tiempo ya que los datos serán de gran volumen y deberán comprimirse. En algunas situaciones, es posible que el histograma de color no funcione correctamente. Por eso, 3]. Los diferentes tipos de técnicas para la extracción de características se analizan en la siguiente sección.

3.3 Sincronización y Computación

La analítica multimedia se ocupa tanto de audio como de video. El principal desafío de la analítica multimedia está en la sincronización de audio y video. Por ejemplo, en los sistemas de salud, los datos multimedia basados en 3-D deben cuidarse para el análisis [8]. Se requiere una sincronización basada en dos niveles para algunos de los sistemas multimedia. La investigación en computación para análisis de datos multimedia involucra muchas plataformas de computación de big data como Hadoop, Cassandra y bases de datos NoSQL. Se requieren tuberías de tales plataformas para el cálculo y el análisis.

Los métodos computacionales de análisis multimedia se basan principalmente en técnicas de aprendizaje de máquinas como máquinas de vectores de soporte (SVM), agrupación. Sin embargo, generalmente se combinan dos o más técnicas para el análisis multimedia. GPU

La informática se utiliza en la mayoría de los análisis multimedia, ya que la cantidad de datos es enorme. La computación en paralelo se lleva a cabo para análisis multimedia con la GPU para reducir el tiempo de computación y tratar los datos multimedia no estacionarios. Sin embargo, los métodos computacionales que son escalables siguen siendo un desafío para la analítica multimedia.

3.4 Seguridad

El análisis multimedia implica datos que están centrados en el usuario y contienen parte de la información confidencial de los usuarios. El análisis realizado sobre dichos datos debe tener las técnicas necesarias para no utilizar la información de identidad del usuario sino realizar análisis [3]. Por ejemplo, en el análisis sentimental de los datos de las redes sociales, no se debe revelar la identidad del usuario, pero se puede utilizar el contenido textual. Aquí, es necesario utilizar el contenido textual en lugar de la identidad de los usuarios. Sin embargo, la analítica geoespacial puede revelar la información geográfica sobre los usuarios. En tales casos, las técnicas analíticas deben tener cuidado al revelar la información basada en el análisis.

4 Tecnologías habilitadoras

El análisis multimedia consiste en analizar datos multimedia como audio, video y texto para identificar patrones en los datos y tomar decisiones basadas en los patrones encontrados. Hay varias etapas de análisis multimedia como preprocesamiento de datos, extracción de características y análisis. En cada etapa, se utilizan diferentes tipos de tecnologías para aprovechar el proceso de análisis y toma de decisiones. Esta sección se centra en las diferentes tecnologías y plataformas habilitadoras para el análisis multimedia. En este sentido, los datos multimedia se clasifican en tres partes, a saber, análisis de texto, análisis de video y análisis de audio. Las tecnologías habilitadoras en cada una de las categorías se resumen como se muestra en la Tabla 1 .

tabla 1 Habilitar tecnologías para multimedia analítica

Tecnología habilitadora	Idioma	Fuente abierta
NLTK	Pitón	sí
Hadoop y Spark	Java, Python, Scala	sí
Cuadro	Java, Python	Parcialmente
Bases de datos de gráficos	Pitón	Parcialmente
LibROSA	Pitón	sí
OpenCV	Pitón	sí

4.1 Análisis de texto

La forma más común de datos multimedia es el texto. Es de naturaleza muy volátil ya que el texto escrito por humanos no está estructurado y difiere de una persona a otra. Hay varias etapas de análisis de texto, como la eliminación de palabras vacías como "es", "era", "esto". Por ejemplo, en el caso del análisis sentimental, las frases clave del texto son importantes para el análisis en lugar de las palabras vacías en el texto. Algunas de las tecnologías habilitadoras que están disponibles para el análisis de texto se analizan a continuación.

- **NLTK**

  El kit de herramientas de lenguaje natural (NLTK) es una de las famosas herramientas basadas en Python que se utilizan para el procesamiento del lenguaje natural. La mayoría de las etapas de análisis de texto, como preprocesamiento, tokenización, derivación, etiquetado, análisis y razonamiento semántico, se pueden realizar utilizando varios módulos que están disponibles dentro de NLTK. La información del corpus es más necesaria al realizar análisis de texto. Por ejemplo, en el escenario del análisis sentimental se necesita un corpus de información que identifique los sentimientos positivos y negativos. WordNet a corpus está disponible en NLTK para análisis de texto. La principal ventaja de NKTK se basa en Python, que ayuda a comprender y analizar fácilmente [9].

- **Aprendizaje profundo**

  La reciente exageración de la inteligencia artificial y la robótica ha allanado el camino para diferentes marcos de aprendizaje profundo. Las plataformas de aprendizaje profundo como Pytorch y Tensor fl ow se utilizan ampliamente para el análisis de texto en sistemas multimedia. Los textos que no se pueden capturar a través de la World Wide Web se analizan utilizando técnicas de aprendizaje profundo como las arquitecturas CNN y RNN. Algunos de los ejemplos incluyen la detección de acciones en video, la identificación de desastres en los conjuntos de datos rastreados en la web.

- **Scikit aprender**

  Es uno de los kits de herramientas científicas disponibles en la plataforma Python. La mayoría de las técnicas de aprendizaje automático se proporcionan en forma de API que ayudan a facilitar la programación. En el análisis de texto, se requiere una bolsa de palabras para identificar las palabras más significativas que están presentes en el conjunto de datos de texto considerado. Scikit learn proporciona funciones como "bolsa de palabras", "tf-idf" que son necesarias para el análisis de texto. El término frecuencia de documentos (TF) se necesita principalmente para identificar los términos más frecuentes en el texto. De esta manera, Scikit learn es una de las tecnologías habilitadoras para el análisis de texto [10].

- **Hadoop y Spark**

  Hadoop es una de las plataformas de código abierto que se puede utilizar para analizar cualquier formato de datos. La programación de MapReduce se utiliza para extraer los datos en forma de texto presentes en el sistema de archivos Hadoop para su análisis. Ayuda a agregar la información que está presente en forma de texto. Por ejemplo, un archivo que contiene los usuarios y los canales suscritos por ellos se puede analizar con la ayuda de MapReduce para encontrar los principales canales suscritos [11]. Spark es una de las plataformas de código abierto que

proporciona numerosas bibliotecas de aprendizaje automático en forma de Spark MLib [12] para realizar diferentes tipos de análisis.

4.2 Análisis de video

La mayor parte de los datos multimedia está en forma de video. El análisis de datos visuales es un proceso complicado, ya que implica una serie de secuencias de imágenes [13]. Aparte de las secuencias de imágenes, los datos visuales pueden estar en forma de gráficos y redes de texto. Hay algunas tecnologías habilitadoras que ayudan en el análisis visual que se resumen a continuación.

- **Redes neuronales de convolución (CNN) mediante aprendizaje profundo**
  CNN es la técnica más utilizada para el análisis de video. Las aplicaciones de CNN al análisis de video incluyen detección de objetos, detección de delitos, etc. En CNN, la secuencia de imágenes se divide primero en varios píxeles y luego se reduce a escala para la detección de objetos. La imagen se divide primero en varios filtros que representan una matriz de valores de la imagen [14]. Los filtros se convierten en una matriz de menor valor por producto y suma de dos matrices. Por ejemplo, una convolución de imagen 2D es como se muestra a continuación.

Imagen 2-D					**Característica convolucionada**		
1 × 0	1 × 1	1 × 0	0 × 1	0 × 0	2	3	2
0 × 1	1 × 0	1 × 1	1 × 0	0 × 0	2	3	3
0 × 1	0 × 1	1 × 0	1 × 1	1 × 0	2	3	3
0 × 0	0 × 1	1 × 1	1 × 0	0 × 1			
0 × 1	1 × 0	1 × 1	0 × 0	0 × 1			

Una vez que se obtiene la matriz de características convolucionada, se construyen una serie de capas de convolución para obtener la matriz de características final y llevar a cabo la detección del objeto. De esta manera, las arquitecturas de CNN se utilizan en análisis de video.

- **Cuadro**
  Los datos multimedia involucran datos de redes sociales que pueden estar en forma de feeds o tweets. La visualización de dichos datos ayuda a revelar los patrones interesantes sobre los datos. Tableau es una de esas herramientas que ayuda a comprender los datos al visualizar primero los datos y luego realizar el análisis. Tableau es la herramienta líder de visualización de datos e inteligencia empresarial que ayuda a crear visualizaciones interactivas. La herramienta de visualización proporciona una interfaz fácil de usar para crear paneles y ayudar a resolver problemas comerciales complejos de una manera rápida y efectiva. Tableau también puede conectarse a un amplio conjunto de archivos, bases de datos, almacén de datos, etc., para realizar análisis de los datos de múltiples fuentes.

- **OpenCV**

  OpenCV es una de las plataformas más utilizadas para análisis de video. Significa biblioteca de visión por computadora de código abierto. Proporciona las bibliotecas necesarias para la detección de objetos en tiempo real dentro de un video. Las aplicaciones de OpenCV incluyen sistema de reconocimiento facial, reconocimiento de gestos, seguimiento de movimiento, realidad aumentada, robótica móvil [15]. Incluye la mayoría de los algoritmos de aprendizaje automático, como el agrupamiento de k-medias, la clasificación bayesiana, el aprendizaje del árbol de decisiones, el clasificador de bosque aleatorio y las redes neuronales artificiales.

4.3 Análisis de audio

Los datos multimedia que se encuentran en forma de audio necesitan una gran cantidad de computación y tiempo para llegar a resultados. Los datos presentes en el formato de audio deben analizarse etapa por etapa, ya que difieren de vez en cuando. Algunas de las tecnologías habilitadoras para el análisis de audio se resumen a continuación.

- **LibROSA**

  Es una de las bibliotecas que está disponible en Python para el análisis de varios archivos de audio [dieciséis]. Ayuda en el desarrollo de varias aplicaciones de audio basadas en diferentes formatos. Ayuda a extraer las características de los archivos de música como ritmo, tempo, tiempos, series de tiempo de audio, espectrograma de potencia del audio, rollo de frecuencia, planitud espectral, etc.

- **Aprendizaje profundo**

  El aprendizaje profundo se utiliza para el análisis de audio utilizando las redes neuronales artificiales, como las redes neuronales recurrentes (RNN), las redes neuronales convolucionales (CNN) y las redes de retroalimentación. Los desarrollos recientes en la informática han allanado el camino para diferentes aplicaciones en el procesamiento de audio mediante el aprendizaje profundo. Uno de esos experimentos se ha realizado en [14] para desarrollar un marco para el reconocimiento de eventos de audio basado en los datos web. ACNN se desarrolla como parte del experimento con cinco capas ocultas. Algunos de los métodos de aprendizaje se exploran en las secciones siguientes.

5 métodos de aprendizaje profundo

En cualquier conjunto de datos, las características / atributos juegan un papel importante en el desarrollo de algoritmos de aprendizaje. En los métodos de aprendizaje profundo, estas características juegan un papel importante en el desarrollo de algoritmos de aprendizaje adecuados. Una característica se puede definir como la parte interesante del conjunto de datos que actúa como punto de partida para el aprendizaje. La propiedad deseable de cualquier algoritmo de aprendizaje es el proceso repetitivo de detectar las características del conjunto de datos que no está entrenado. Las características importantes de un dataset de imágenes son temporales, espaciales y de textura. Hay tres etapas en cualquier algoritmo de detección de características:

extracción, selección y clasificación. El resultado de la fase de extracción es un vector de representación del conjunto de datos / imagen considerado. Los vectores de representación pueden ser diferentes para diferentes conjuntos de datos. Se utiliza con fines de clasificación mediante métodos de aprendizaje profundo como CNN, RNN y redes de alimentación directa [17 , 18].

5.1 Métodos de extracción de características locales

Una característica es una función que representa el valor cuanti fi cado de un objeto / conjunto de datos. Representa las características importantes del objeto, como el color, la textura y los píxeles. Las características locales de un conjunto de datos / imagen se refieren a las características de la imagen que se adhiere a la detección de bordes, la segmentación de la imagen y las características que se calculan sobre los resultados de la subdivisión de la imagen. Los métodos más comunes que se utilizan para la extracción de características locales mediante el aprendizaje profundo son los siguientes.

- **Detección de esquinas Haris**
  La detección de esquinas se utiliza en los sistemas de visión por computadora para inferir algunos de los datos interesantes de la imagen y sus esquinas. Por lo general, se usa para aplicaciones como detección de movimiento, seguimiento de video, modelado 3D y reconocimiento de objetos. Uno de los métodos que se utilizan es la detección de esquinas de Haris que se basa en la intensidad de las esquinas [19]. La idea básica es observar un gran cambio en la apariencia cuando hay un desplazamiento en la ventana en cualquier dirección. Los tres escenarios normales de detección de esquinas son los que se muestran en la Fig. 2 . El cambio en el desplazamiento de la intensidad dice [u, v] se calcula como se muestra en la Eq. 1 .

- **Transformación de características invariantes de escala (SIFT)**
  Se utiliza para detectar características locales en la imagen para aplicaciones tales como reconocimiento de gestos, seguimiento de video, modelado 3D, etc. Un conjunto de características clave en la imagen se extraen y almacenan en una base de datos. Para una nueva imagen, se comparan las características

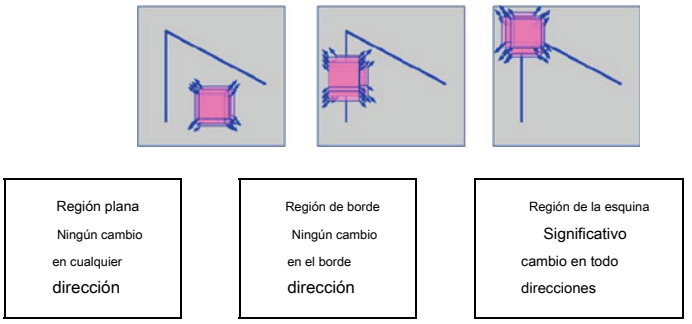


Figura 2 Detección de esquinas Haris

contra la base de datos existente para determinar la mejor coincidencia de candidatos utilizando la distancia euclidiana de los vectores de características [20]. Para filtrar las coincidencias de la base de datos, se filtra de la base de datos un subconjunto de los puntos clave que coinciden con la ubicación, la escala y la orientación. El inconveniente de SIFT es que el cálculo implica el cálculo del valor del parche de píxeles basado en el histograma de gradientes. Por tanto, si el tamaño de la imagen es grande, el cálculo se vuelve caro.

$$UE, v) = \sum_{x, y} w (x, y) [I (x + u, y + u) - Yo (x, y)]_2 \quad (1)$$

- **BRIEF orientado rápido y girado (ORB)**

ORB es uno de los métodos de extracción de características locales que supera el inconveniente de SIFT y reduce el tiempo de cálculo. Se basa en el BRIEF (características elementales independientes robustas binarias) y el detector de puntos FAST [21]. Los puntos se detectan en la imagen según la orientación. Inicialmente, el punto se ubica en el centro de la esquina de la imagen. La orientación de la imagen se conoce a partir de este punto. La invariancia se mejora usando la región circular de la imagen. La descripción de las imágenes se establece utilizando los puntos detectados mediante el detector de puntos FAST. Una matriz S (2 × n) de fi ne las coordenadas de los píxeles del conjunto de características. La orientación

encontrado usando el punto FAST gira la matriz S a una dirección

versión S. Primero, se calcula previamente una tabla de búsqueda utilizando el ángulo 2/30 ° para producir

el conjunto correcto de puntos S. Un pequeño fragmento del código se muestra a continuación.

La imagen se lee inicialmente ingresada en escala de grises, pero se puede convertir a RGB más tarde.

El extractor de características locales ORB está de fi nido y especifica el número de características que se necesitan para recoger. Inicialmente, se establece en las 1000 características principales, si se deja en blanco si encuentra todas las características posibles. Todos los puntos de interés de la imagen se calculan y almacenan en la variable “kp”. Se define un gráfico para estos puntos característicos almacenados en "kp" en la imagen original, estos puntos están marcados en un solo color como se especifica en la lista de parámetros. Al final, la salida se puede ver usando la función imshow () en matplotlib. La salida de una imagen de muestra es como se muestra en la Fig. 3 .

Imagen de entrada Funciones de ORB

Fig. 3 Salida de muestra ORB

```
importar cv2
importar matplotlib.pyplot como plt
filename = '5.jpg' #filename of the image img = cv2.imread ('5.jpg', 0)

orb = cv2.ORB (nfeatures = 1000) #define cuántas características elegir

kp = orb.detect (img, Ninguno)
kp, des = orbe.compute (img, kp)
img2 = cv2.drawKeypoints (img, kp, color = (255,0,0), banderas = 0)
# trama todos la puntos de interesar en la imagen
plt.figure (figsize = (16, 16))
plt.title ('Puntos de interés ORB')
plt.imshow (img2) #imprimir imagen con resaltados plt.show ()
```

5.2 Métodos de extracción de características globales

Una vez que las características locales, como los bordes de las imágenes / conjunto de datos, se extraen de la imagen, las características globales se calculan para toda la imagen. Estas características globales ayudan en la clasificación final de la imagen utilizando los elementos calculados de las características locales. Los métodos comunes que se utilizan para extraer características globales del conjunto de datos / imagen son los siguientes:

- **Histograma de gradientes orientados (HOG)**
  HOG se utiliza como descriptor de funciones en el procesamiento de imágenes y la visión por computadora para la detección de objetos. Se basa en el histograma de gradientes que actúa como una característica global para la detección de objetos. En partes locales de la imagen, cuenta las ocurrencias de orientación de gradiente para producir el histograma. La intensidad de la distribución de los histogramas presentes en las apariencias de los objetos locales guía la producción de los histogramas [22]. Básicamente, en una imagen, cada píxel se dibuja en un grupo de celdas. Se compila un histograma de degradados para cada píxel de la celda. Un descriptor forma la concatenación de todos estos histogramas. Los diferentes tipos de gradientes que se siguen en HOG se enumeran a continuación.

  - **Degradados de imagen**
    Un gradiente es una cantidad vectorial que tiene magnitud y dirección. Un degradado de imagen da el cambio de intensidad en la ubicación particular de la imagen. Un gradiente de imagen ilustrativo es como se muestra en la Fig. 4 .
  - **Histograma de gradientes orientados**
    Cuando los degradados de la imagen están orientados, se denomina degradados orientados. Un histograma de gradientes orientados da la intensidad de los gradientes del

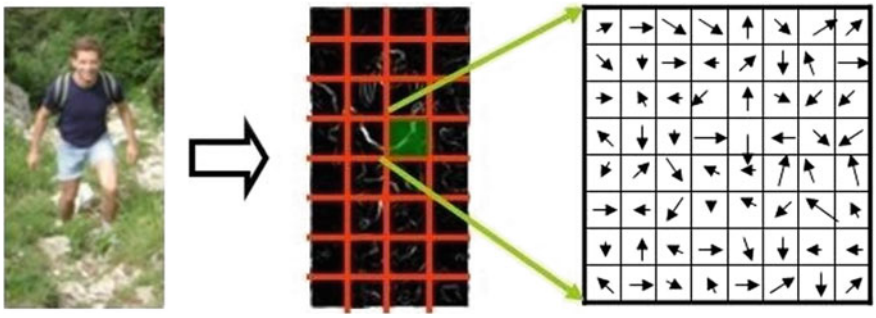


Figura 4 Gradiente de imagen

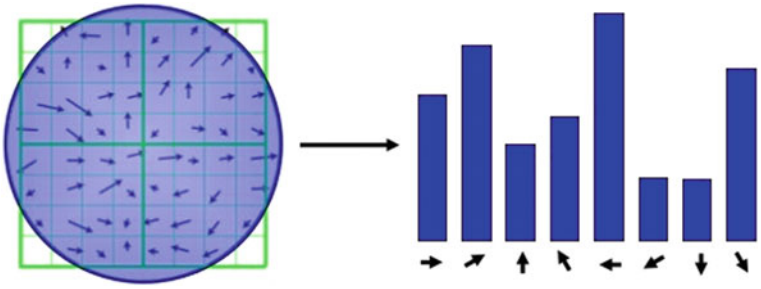


Figura 5 Histograma de gradientes orientados

pequeña porción de la imagen. Ayuda a encontrar la probabilidad de un gradiente en la parte particular de la imagen como se muestra en la Fig. 5 .

Un pequeño fragmento de Histograma de gradientes orientados se muestra a continuación. Se importan las bibliotecas necesarias, incluida la función de detector HOG de la biblioteca de skimage. Aquí, img es la imagen que se está leyendo, la orientación es el número de direcciones para las características en términos de fluctuación de intensidad, pixels_per_cell divide la imagen en celdas y de fi ne el número de píxeles por celda, cells_per_block es el número de celdas tomadas por bloque para mapear, visualizar es obtener una imagen final de la detección de características, feature_vector es para generar el vector de características final, "feat" es el conjunto de características e "im" es el histograma. La salida de una imagen de muestra es como se muestra en la Fig. 6 C.A.

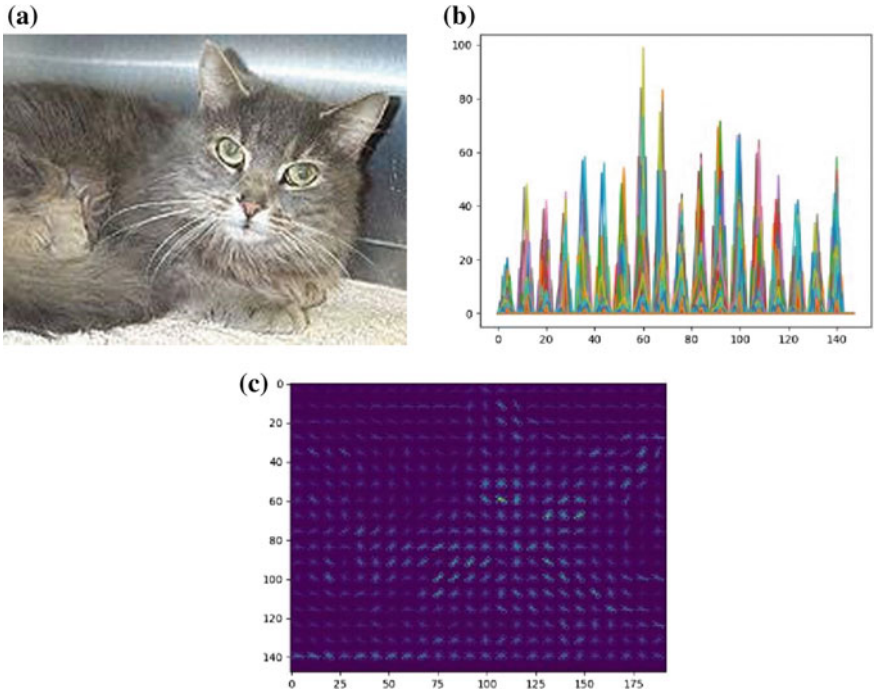


Figura 6 a Entrada para HOG **B** Histograma para la salida de muestra de HOG **C** Funciones para la salida de muestra de HOG

```
importar cv2
importar matplotlib.pyplot como plt desde
skimage.feature import hog
img = cv2.imread ("7.jpg", 0) #leer imagen
feat, im = hog (img, orientations = 9, pixels_per_cell = (8,8), cell s_per_block = (2,2), visualize =
True, feature_vector = True)
# img es la imagen que se está leyendo
# orientación el número de direcciones para las características en términos de fluctuación
 de intensidad

# pixels_per_cell divide la imagen en celdas y define la densidad de píxeles


# cells_per_block número de celdas tomadas por bloque para mapeo


# visualizar obtener imagen de detección de características
# feature_vector
plt.plot (im) #plot histograma para detección plt.show () #show
histpgram graph
```

6 Aprendizaje profundo para la clasificación de datos de IoT

6.1 Datos de IoT y sus tipos

Internet de las cosas se compone de varios dispositivos capaces de recibir datos y pueden poseer o no poder de cómputo. Estos dispositivos están interconectados y generalmente están en constante comunicación entre sí. Así, como consecuencia natural, siempre sigamos generando datos. Los dispositivos que participan en una red de IoT se pueden clasificar principalmente en dos tipos. El primero de ellos que forma los bloques de construcción de la red son los sensores y el segundo de ellos son los dispositivos que consumen los datos adquiridos de los sensores. Estos dispositivos que consumen los datos se pueden distinguir además como los que realizan análisis basados en los datos recuperados o inician una acción basada en los datos adquiridos. En general, los datos de IoT que generan los sensores desempeñan un papel importante en la potencia de varios tipos de aplicaciones.

A partir de la comprensión natural del comportamiento de los dispositivos que participan en IoT, se puede derivar una norma para clasificar los datos de IoT. Por lo tanto, esta clasificación básica de los datos de IoT produce dos categorías, a saber

1. Datos utilizados con fines de análisis (datos de análisis)
2. Datos utilizados para iniciar una acción (datos de actuación o datos que provocan una acción)

6.1.1 Datos de análisis

El volumen de datos generado a partir de una amplia gama de sensores proporciona una plataforma excelente para realizar análisis. Hay varios tipos de datos de sensores disponibles que van desde datos triviales obtenidos de sensores de movimiento simples hasta una forma más sofisticada de datos, como datos de ubicación obtenidos de teléfonos inteligentes. Los datos obtenidos generalmente estarán en forma de datos de series de tiempo, lo que tiene la consecuencia natural de tener un mayor volumen.

Un ejemplo clásico que puede ilustrarse en el caso del análisis de datos de IoT es el del estudio de datos meteorológicos. Consiste en el estudio de la atmósfera y sus componentes y varios análisis relacionados, como patrones climáticos, magnitudes de lluvia, etc. También se realizan varias predicciones basadas en estos datos, que son claramente evidentes en forma de actualizaciones meteorológicas, predicciones de lluvia y sugerencias de rango de temperatura. Otro ejemplo similar será el análisis de los componentes químicos del aire y el análisis de la calidad general del aire.

Varios otros dominios también brindan un buen alcance para el análisis de datos que incluye fuentes multimedia. Este vasto dominio tiene una enorme disposición de datos que existen en varias formas. Esto incluye datos de tráfico en tiempo real, texto e imágenes de varias plataformas de medios sociales. Otro dominio que proporciona una plataforma para obtener estadísticas son los diversos enrutadores, conmutadores y otros dispositivos de red implementados en todo el mundo. Los sistemas SDN modernos tienen estadísticas de dispositivos integradas para permitir el análisis e inferir niveles de congestión y otros parámetros similares.

Los equipos también participan en la generación de datos tales como electrocardiografía, informes de análisis de sangre, rayos X, etc., que pueden utilizarse con fines de análisis cuando se ven a gran escala. Los registros de llamadas de teléfonos móviles y los tiempos de uso de diferentes aplicaciones también pueden considerarse como un contribuyente. La información obtenida después de realizar el análisis se utiliza generalmente con fines comerciales o puede ser utilizada por el gobierno.

6.1.2 Datos de actuación o datos de causa de acción

Además de ayudar a varias aplicaciones de análisis, IoTdata también se utiliza para activar otros dispositivos con el fin de obtener una acción deseable o un mecanismo que impulse una decisión. Esta forma de utilidad se está explotando en áreas como los sistemas de automatización y en escenarios de redes, como un mecanismo de solución para problemas de congestión.

Por ejemplo, en el caso de los sistemas de automatización, un sistema de domótica simple sirve para comprender los mecanismos de toma de decisiones. Los sistemas de calefacción e iluminación presentes en una casa típica, como una bombilla normal o un ventilador, se pueden alternar entre el estado de encendido y apagado en función de los datos recopilados por los sensores implementados en la casa, que incluyen sensores de movimiento, termostatos y sensores relacionados.

Como ejemplo, en el caso de escenarios de redes, se puede recuperar la información de enrutamiento almacenada en los dispositivos periféricos y los registros de datos o paquetes que pasan a través del dispositivo. Esta información se puede utilizar para permitir el reencaminamiento, lo que produce el equilibrio de carga y, por lo tanto, permite el control de la congestión.

6.2 Estudio de caso sobre la clasificación de datos de IoT mediante el aprendizaje profundo

6.2.1 Conjunto de datos de IoT

El conjunto de datos elegido para fines de clasificación es "Conjunto de datos de calidad del aire" [23]. Es un conjunto de datos de tamaño considerable de 9358 instancias. La calidad del aire está determinada por la composición relativa de sus elementos constituyentes. Los datos recopilados aquí se pueden clasificar como datos de IoT, ya que los han acumulado los distintos sensores. El dispositivo utilizado para formar el conjunto de datos es un dispositivo multisensor químico de calidad del aire. Está compuesto por cinco sensores químicos de óxido metálico que están dispuestos para formar el dispositivo.

Los datos se recopilaron por horas para obtener un conjunto de datos considerable y que representa una variación adecuada. El sensor se colocó en una ciudad italiana que estaba sujeta a un nivel relativamente más alto de contaminación debido a la presencia de varios contaminantes. La duración de la captura de datos varió desde marzo de 2004 hasta febrero de 2005. Estos datos representan un año completo de registro, que se considera la duración más larga en la que se instalaron sensores químicos que capturaron la calidad del aire.

El conjunto de datos consta de los siguientes atributos:

1. Fecha (DD / MM / AAAA)
2. Hora (HH.MM.SS)
3. Concentración promedio por hora real de CO en mg / m̂ 3 (analizador de referencia) PT08.S1 (óxido de estaño)
4. Respuesta del sensor promediada por hora (nominalmente objetivo de CO) Concentración de hidrocarburos no
5. metánicos general promedio por hora real en microg / mˆ3 (analizador de referencia)
6. Concentración de benceno promedio por hora real en microg / m / 3 (analizador de referencia)
7. PT08.S2 (titania) Respuesta del sensor promediada por hora (nominalmente objetivo de NMHC) Concentración de NOx
8. promediada por hora real en ppb (analizador de referencia)
9. PT08.S3 (óxido de tungsteno) respuesta del sensor promediada por hora (nominalmente dirigido a NOx)
10. Verdadero promedio horario NO_2 concentración en microg / m̂ 3 (analizador de referencia)
11. PT08.S4 (óxido de tungsteno) respuesta del sensor promediada por hora (nominalmente NO_2 dirigido)
12. PT08.S5 (óxido de indio) respuesta del sensor promediada por hora (nominalmente O_3 dirigido)
13. Temperatura en ° C
14. Humedad relativa (%)
15. Humedad absoluta AH

La mayoría de las veces, los datos de IoT se capturan con la ayuda de varios sensores. Estos sensores suelen ser físicos y se implementan en los sitios. También se pueden simular a través de un script de Python con fines experimentales o de análisis. Estos scripts que se implementan en dispositivos de borde actúan como una interfaz para simular diferentes tipos de sensores. La secuencia de comandos que se muestra a continuación se puede utilizar para bombear datos del sensor.

Los sensores implementados en los dispositivos periféricos necesitan bombear datos a una base de datos central. Los dispositivos de IoT se comunican mediante un protocolo ligero llamado protocolo mqtt. Se publican y suscriben dos operaciones o funciones básicas asociadas con el protocolo mqtt. Estos términos son análogos a la escritura y la lectura de un dispositivo de E / S. Un dispositivo o un sensor que necesita bombear datos tiene que publicar con un tipo particular bajo un tema predeterminado. Del mismo modo, los dispositivos que necesitan analizar o leer estos datos deben suscribirse al tema. El corredor de mqtt juega un papel vital o es esencialmente el corazón del protocolo. Es responsable de enviar y recibir mensajes y entregarlos a los clientes, por lo que es responsable del modelo fundamental de publicación / suscripción. En el script que se muestra a continuación, los datos deben publicarse y, por lo tanto, se ha incorporado el código asociado.

```
tiempo de importación
importar paho.mqtt.client como mqtt import json
import ssl
importar al azar
broker = "<broker_url here>" port = 8883

topic = "iot-2 / evt / test / fmt / json"
username = "<nombre de usuario>"
contraseña = "<contraseña>" organización = "<id
de la organización aquí>" deviceType =
"Python_Client"

mqttc = mqtt.Client ("<id de cliente>") print (nombre de usuario + " + contraseña)

mqttc.username_pw_set (nombre de usuario, contraseña)
mqttc.tls_set_context (contexto = Ninguno)
mqttc.tls_insecure_set (falso)
mqttc.connect (corredor, 8883,60)
  para i inrange (0,10):
                print ('Enviando ejemplos de pares clave-valor') msg   = {'clave': i, 'valor':
               random.randrange (1,50)}
               mqttc.publish (tema, json.dumps (msg))
               hora de dormir (1)
```

Se han importado las bibliotecas de soporte para implementar el protocolo mqtt y establecer un canal seguro para la comunicación. Se ha elegido un número de puerto adecuado y el tipo de dispositivo deseado. Entre los diversos corredores mqtt disponibles, se debe elegir un corredor apto según la aplicación. El script anterior simula un sensor de muestra y, por lo tanto, se toma un par clave-valor genérico como lectura del sensor y se elige un nombre de tema de muestra. Los pares clave-valor generalmente denotan el (período de tiempo, valor del sensor) de un sensor típico. Este sensor genérico se puede considerar como un indicador de temperatura o un acelerómetro, en términos simples, un sensor de gravedad que está presente en la mayoría de los teléfonos inteligentes modernos. Estos sensores virtuales se pueden utilizar con fines de análisis de muestras y para adaptarse a los requisitos de la vida real.

6.2.2 Construcción de una red neuronal para la clasificación

Muchos problemas como correo no deseado, reconocimiento de dígitos de escritura a mano, detección de transacciones fraudulentas son ejemplos importantes del día a día de problemas de clasificación. La clasificación en términos simples, es la agrupación de cosas por características, características y cualidades comunes. Para resolver este problema existen muchos algoritmos de clasificación como árbol de decisión, regresión logística y k vecinos más cercanos. Sin embargo, Neural Network tiene pocos beneficios en comparación con otras, por lo que es la opción más popular para manejar big data en tiempo real. Los beneficios de la red neuronal son:

- El algoritmo de aprendizaje automático normal llegará a una meseta y no mejorará mucho con más datos después de cierto punto. Sin embargo, la red neuronal mejora con más datos para entrenar

- Neural Network es un modelo no lineal, que nos ayuda a definir límites de decisión complejos, no triviales, de hiperplanos para resolver problemas de clasificación complicados.

- La red neuronal es altamente adaptable y puede hacer frente a entornos no estacionarios.
- La red neuronal tiene una alta tolerancia a fallas. La corrupción de algunas neuronas no le impide generar la salida.
- Cada neurona tiene el potencial de influir en todas las demás neuronas de la red. Por tanto, la información contextual se trata de forma natural.

Debido a todas las razones anteriores, la red neuronal artificial se está empleando activamente en muchos problemas del mundo real como clasificar correos electrónicos no deseados, predicción de quiebras, marcar transacciones fraudulentas, etc., y están obteniendo mejores resultados que los algoritmos convencionales.

6.2.3 Experimento y resultados

Un conjunto de datos de muestra se muestra en la Tabla 2 . La red neuronal artificial es una red de neuronas. Procesan los datos de entrenamiento uno a la vez y aprenden comparando su clasificación predicha con la clasificación real. Los errores de la clasificación inicial se retroalimentan a la red para corregirse y modificar los pesos para una iteración posterior. Por lo tanto, la principal dificultad en la construcción de una red neuronal es encontrar la agrupación más apropiada de funciones de entrenamiento, aprendizaje y transferencia para obtener el mejor resultado.

Antes de que podamos construir una red, procesamos el conjunto de datos clasificando todas las columnas de NO2 (GT) en dos clases. 1 para mayor y 0 para menos de 100 índice de calidad del aire como se muestra en la Tabla 3 .

Tabla 2 Conjunto de datos de calidad del aire

Fecha	Hora	CO (GT)	PT08.S1 (CO)	NMHC (GT)	C6H6 (GT)	PT08.S2 (NMHC)
10-03-2004	18.00.00	2.6	1360	150	11,9	1046
10-03-2004	19.00.00	2	1292	112	9.4	955
10-03-2004	20.00.00	2.2	1402	88	9	939
10-03-2004	21.00.00	2.2	1376	80	9.2	948
10-03-2004	22.00.00	1,6	1272	51	6.5	836

Tabla 3 Preprocesamiento de datos

Fecha	Hora	NO2 (GT)	T
10-03-2004	18.00.00	1	13,6
10-03-2004	19.00.00	0	13,3
10-03-2004	20.00.00	1	11,9
10-03-2004	21.00.00	1	11,0
10-03-2004	22.00.00	1	11,2

```
data = df [['Fecha', 'Hora', 'NO2 (GT)', 'T']]. copy ()
para i en rango (len (datos)):

    si (data.iloc [i] ['NO2 (GT)'] <100):

        data.at [i, 'NO2 (GT)'] = 0 más:
        data.at [i, 'NO2 (GT)'] = 1
```

A continuación, necesitamos características de entrada como el día de la semana y la hora para extraer de la columna de fecha y hora como se muestra en la Tabla 4 . Podemos dividir el conjunto de datos en conjunto de entrenamiento y conjunto de prueba.

```
para i en rango (len (datos)):
    time = datetime.datetime.strptime (data.iloc [i] ['Fecha'
     ] + "" + data.iloc [i] ['Hora'], "% d /% m /% Y% H:% M:% S") día de la semana = hora.día
    de la semana ()
    hora = hora.hora
    data.at [i, 'Weekday'] = día de la semana
    data.at [i, 'hour'] = hora
    data.Weekday = data.Weekday.astype (int)
    data.hour = data.hour.astype (int)
    train, test = train_test_split (data, test_size = 0.2) train.to_csv ("train.csv", mode = "w",
    index = False)
    test.to_csv ("test.csv", mode = "w", index = False)
```

Cuadro 4 Preprocesamiento de datos con fecha de día y hora

	Hora	NO2 (GT)	T	Día laborable	Hora
10-03-2004	18.00.00	1	13,6	2	18
10-03-2004	19.00.00	0	13,3	2	19
10-03-2004	20.00.00	1	11,9	2	20
10-03-2004	21.00.00	1	11,0	2	21
10-03-2004	22.00.00	1	11,2	2	22

Usaremos una API de alto nivel de flujo tensorial para construir una red neuronal simple con una capa de entrada (4 nodos) y una capa de salida. Para eso, primero tenemos que escribir una función para extraer el conjunto de datos de entrenamiento.

```
def load_traindata (label_name = 'NO2 (GT)'):
    train_path = "train.csv" # Para el modelo NN de entrenamiento
    CSV_COLUMN_NAMES =
    ['Fecha', 'Hora', 'NO2 (GT)', 'T', 'Día de la semana', 'hora']
    train = pd.read_csv (filepath_or_buffer = train_path,
    names = CSV_COLUMN_NAMES, # lista de nombres de columna header = 0, #
    ignora la primera fila del archivo CSV.
        skipinitialspace = True,
        # saltos = 1
        )
    train.pop ('Hora')
    train.pop ('Fecha')
    tren ['hora'] = tren ['hora']. astype (str)
    train.Weekday = tren.Weekday.astype (str)
    tren ['T'] = tren ['T']. astype (flotar)
    train_features, train_label = tren, tren.pop (label_n
    ame)
    return (características del tren, etiqueta del tren)

# obtener funciones y etiquetas de capacitación
(función_entrenamiento, etiqueta_transporte) = load_traindata ()
```

Para darle características de entrada, tenemos que darlo en forma de columna tf.feature. Esta característica puede ser categórica (día de la semana, hora) o numérica (temperatura T) también. Si dos o más funciones de entrada están estrechamente relacionadas, puede ser una función separada en sí misma. Este tipo de columnas se conoce como columna cruzada (día de la semana x_hora), que es una combinación de dos o más características de entrada. Después de decidir las características de entrada, creamos un clasificador SimpleDNN con columnas base y columnas cruzadas. Hiddenunit indica el número de nodos en cada capa (por ejemplo [3 - 5] indica una red neuronal con capa de entrada con 4 nodos, capa oculta con 3 nodos y capa oculta 2 con 2 nodos). Necesitamos una capa oculta si los datos no se pueden separar linealmente. El número de nodos de entrada depende del número de características de entrada y, de manera similar, el número de nodos de salida depende del número de clases en la etiqueta de salida.

```
# crear características normales y características cruzadas para el modelo nn

Día de la semana =
tf.feature_column.categorical_column_with_vocabulary_li st ('Día de la semana', ['0', '1', '2',
'3', '4', '5', '6'])
hora =
tf.feature_column.categorical_column_with_vocabulary_li st ('hora', ['0', '1', '2', '3', '4', '5', '6',
'7', '8', '10 ',' 12 ',' 13 ',' 14 ',' 15 ',' 16 ',' 17 ',' 18 ',' 19 ',' 20 ',' 21 ',' 22 ',' 23 '] )


T =
tf.feature_column.numeric_column (clave = 'T', dtype = tf.float
64)
base_columns = [
        tf.feature_column.indicator_column (día de la semana),
        tf.feature_column.indicator_column (hora),
        T
        ]
Weekday_x_hour = tf.feature_column.crossed_column (
   ['Día de la semana', 'hora'], hash_bucket_size = 1000)
columnas_cruzadas = [
   tf.feature_column.indicator_column (Weekday_x_hour)
        ]
# Ejecución del modelo de clasificador Dnn con las características diseñadas

# Arriba y con una capa de entrada con 4 nodos

clasificador =
tf.estimator.DNNClassifier (feature_columns =
columnas_base + columnas_cruzadas, unidades_ ocultas = [4], n_
clases = 2)
```

Tenemos que entrenar al clasificador pasando una función train_input que cambia las características de datos de entrenamiento de entrada. Tenemos que iterar a través del proceso (época) hasta que el clasificador alcance su tasa de error mínima. Aquí hemos optado por iterar 50 veces.

```
# entrenando el nnmodel
def train_input_fn (características, etiquetas, tamaño de lote):
        conjunto de datos =
        tf.data.Dataset.from_tensor_slices ((dict (características),
        etiquetas))
        conjunto de datos =
        dataset.shuffle (buffer_size = 1000) .repeat (count = None
         ). lote (tamaño_lote)
       regreso
       dataset.make_one_shot_iterator (). get_next ()
```

```
classifier.train
    input_fn = lambda: train_input_fn (train_feature,
    train_label, 50), pasos = 1000)
```

Entrenamiento de redes neuronales

```
INFO: tensorflow: Callingmodel_fn.
INFO: tensorflow: Terminado de llamar model_fn.
INFO: tensorflow: CreateCheckpointSaverHook.
INFO: tensorflow: se finalizó el gráfico.
INFO: tensorflow: Runninglocal_init_op.
INFO: tensorflow: Terminado de ejecutar local_init_op.
INFO: tensorflow: guardando puntos de control para 1 en
C: \ Users \ Guest \ AppData \ Local \ Temp \ tmprvwl47wc \ model.ckpt.
INFO: tensorflow: paso = 1, pérdida = 42.49231
INFO: tensorflow: global_step / sec: 48.6003
INFO: tensorflow: paso = 101, pérdida = 30.568665 (2.062 seg)

INFO: tensorflow: global_step / sec: 55.7494
INFO: tensorflow: paso = 201, pérdida = 25.75341 (1.793 seg) INFO: tensorflow: global_step
/ seg: 57.967
INFO: tensorflow: paso = 301, pérdida = 24.957882 (1.726 seg) INFO: tensorflow:
global_step / seg: 57.7436
INFO: tensorflow: paso = 401, pérdida = 29.967522 (1.732 seg) INFO: tensorflow:
global_step / seg: 58.4679
INFO: tensorflow: paso = 501, pérdida = 27.571487 (1.711 seg)

INFO: tensorflow: global_step / sec: 53.1789
INFO: tensorflow: paso = 601, pérdida = 25.81527 (1.875 seg) INFO: tensorflow: global_step
/ seg: 54.6941
INFO: tensorflow: paso = 701, pérdida = 19.551216 (1.833 seg) INFO: tensorflow:
global_step / seg: 56.3506
INFO: tensorflow: paso = 801, pérdida = 27.794727 (1.776 seg) INFO: tensorflow:
global_step / seg: 59.6893
INFO: tensorflow: paso = 901, pérdida = 21.918167 (1.673 seg)

INFO: tensorflow: guardando puntos de control para 1000 en
    C: \ Users \ Guest \ AppData
    \ Local \ Temp \ tmprvwl47wc \ model.ckpt.
INFO: tensorflow: Pérdida para el paso final: 24,991734.
```

Después de entrenar al clasificador, podemos probarlo pasándole las características de entrada de datos de prueba, de manera similar a como lo entrenamos. Tenemos que pasar una función que evalúe la función de entrada para pasar sólo la característica de entrada al clasificador y el clasificador predice la salida.

```
    # predecir para todos los intervalos de tiempo en un día con el capacitado
                                                                                modelo
test_df = pd.read_csv ("test.csv", parse_dates = True)
predecir_x = {
            'T': [],
            'Día laborable':[],
            'hora':[ ],
            }
 predecir_x ['T'] = test_df ['T']. astype (flotante)
 predecir_x ['hora'] = test_df.hour.astype (str)
 predecir_x ['Día de la semana'] = test_df.Weekday.astype (str)
 test_label = test_df ['NO2 (GT)']

 def eval_input_fn (características, etiquetas = Ninguna, tamaño de lote = Ninguna):
      "" "Una función de entrada para evaluación o predicción" "" si las etiquetas son Ninguna:

                  # Sin etiquetas, use solo funciones. entradas =
                    características

      demás:
                  entradas = (características, etiquetas)
                  # Convierta las entradas en un objeto tf.dataset.

                  conjunto de datos = tf.data.Dataset.from_tensor_
                  rodajas (entradas)
                  # Lote de ejemplos
                  afirmar batch_size no es None, "batch_size no debe ser None"

                  conjunto de datos = conjunto de datos.batch (tamaño_de_lote)
                  # Devuelve el final leído de la canalización.
                  regreso
                  dataset.make_one_shot_iterator (). get_next ()

 predicciones = clasificador.predicto (
 input_fn = lambda: eval_input_fn (predecir_x,
          etiquetas = Ninguno, tamaño_lote = 50))

 pred_label = []
 para pred_dict en zip (predicciones):
      si pred_dict [0] ['clases'] == [b'1 ']:
               pred_label.append (1)
      demás:
              pred_label.append (0)
```

Dado que el problema elegido es una clasificación binaria, podemos verificar los resultados encontrando la matriz de confusión para los datos de prueba. La matriz de confusión es una tabla que contiene verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos como se muestra en la tabla. 5 .

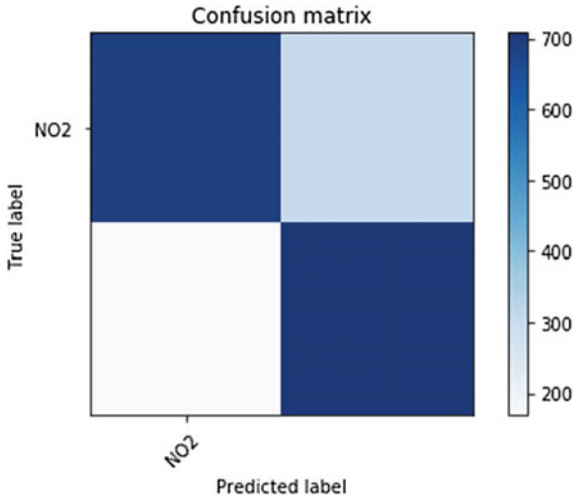
Cuadro 5 Matriz de confusión

	Positivo previsto	Negativo previsto
Positivo real	692	303
Negativo real	169	708

Obtenemos la sensibilidad y la especificidad con las que el clasificador clasifica los datos de prueba. También podemos representar la matriz de confusión en forma de gráfico como se muestra en la Fig. 7 .

```
# trazar una matriz de confusión con los datos probados def plot_confusion_matrix (cm,
                                                    nombres          title = 'Confusión
matriz ', cmap = plt.cm.Blues):
plt.imshow (cm, interpolación = 'más cercano', cmap = cmap)
plt.title (título)
plt.colorbar ()
tick_marks = np.arange (len (nombres))
plt.xticks (tick_marks, nombres, rotación = 45)
plt.yticks (tick_marks, nombres)
plt.tight_layout () plt.ylabel ('Etiqueta verdadera')
plt.xlabel ('Etiqueta prevista')
con_mat = tf.confusion_matrix (test_label, pred_label)
cm = []
con tf.Session ():
    cm = tf.Tensor.eval (con_mat, feed_dict = None,
     session = Ninguno)
```

Figura 7 Representación visual de la matriz de confusión



```
plt.figure ()


diagnóstico = ["NO2"]
 plot_confusion_matrix (cm, diagnóstico)
```

También podemos probar nuestros resultados de nuestro clasi fi cador usando una curva ROC (Receiver Operating Characteristic). La curva ROC es una forma gráfica de mostrar el límite entre la sensibilidad (fracción de verdaderos positivos) y la especificidad (fracción de verdaderos negativos). El área bajo la curva Roc es una medida de la utilidad de la prueba, por lo tanto, un AOC mayor significa un mejor resultado como se muestra en la Fig. 8 .

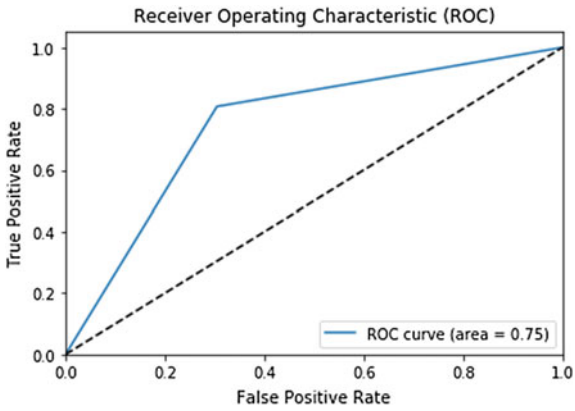
```
# Trace una ROC. pred - las predicciones, y - el resultado esperado. def plot_roc (pred, y):

fpr, tpr, _ = roc_curve (y, pred)
roc_auc = auc (fpr, tpr)
print ("Auc de clasifer es") print (roc_auc)

plt.figure ()
plt.plot (fpr, tpr, label = 'Curva ROC (área =% 0.2f)'
                % roc_auc)
plt.plot ([0, 1], [0, 1], 'k--') plt.xlim ([0.0, 1.0])

plt.ylim ([0.0, 1.05])
plt.xlabel ('Tasa de falsos positivos')
plt.ylabel ('Tasa de verdaderos positivos')
plt.title ('Característica de funcionamiento del receptor (ROC)')
plt.legend (loc = "abajo a la derecha")
plt.show ()
plot_roc (pred_label, test_label)
```

Figura 8 Representación visual de la curva ROC



7. Conclusión

Dado que los dispositivos de IoT aumentan día a día, los datos multimedia de diferentes sensores, como video, audio, teléfono y otros, deben procesarse en tiempo real. Si los datos generados no se recopilan y analizan, es posible que no se conozca el valor de los datos. Por lo tanto, los métodos de aprendizaje profundo son muy esenciales para el análisis de datos multimedia e IoT para diferentes aplicaciones. En este capítulo, se mencionó inicialmente una breve introducción a los datos multimedia y sus tipos. Los diferentes métodos de aprendizaje profundo con extracción de características locales y extracción de características globales se discutieron con pequeños fragmentos de ejemplos. Finalmente, un estudio de caso sobre monitoreo de la calidad del aire presentó el método de aprendizaje profundo de clasificación y análisis de los datos.

Referencias

1. A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. Maasberg, K.-KR Choo, Aplicaciones multimedia de big data y Internet de las cosas: una taxonomía y un modelo de proceso. J. Netw. Computación Appl. (2018)
2. PK Atrey, M. Anwar Hossain, A. El Saddik, MS Kankanhalli, Fusión multimodal para análisis multimedia: una encuesta. Multimed. Syst. **dieciséis(** 6), 345–379 (2010)
3. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet de las cosas (IoT): una visión, elementos arquitectónicos y direcciones futuras. Future Gener. Computación. Syst. **29 (** 7), 1645–1660 (2013)
4. CA Bhatt, MS Kankanhalli, Minería de datos multimedia: estado del arte y desafíos. Multimed. Herramientas Appl. **51 (** 1), 35–76 (2011)
5. F. Venter, A. Stein, Imágenes y vídeos: datos realmente grandes. Anal. revista 14 a 47 (2012)
6. D. Che, M. Safran, Z. Peng, De big data a big data mining: desafíos, problemas y oportunidades, en *Sistemas de bases de datos para aplicaciones avanzadas (* Springer, Wuhan, China, 2013), págs. 1-15
7. Z. Wu, M. Zou, Un método de detección de comunidad incremental para sistemas de etiquetado social que utilizan hash sensible a la localidad. Neural Netw. **58 (** 1), 12 a 28 (2014)
8. PK Atrey, NC Maddage, MS Kankanhalli, detección de eventos basada en audio para vigilancia multimedia, en *2006 IEEE International Conference on Acustics Speech and Signal Processing Proceedings,* vol. 5 (IEEE, 2006)
9. NLTK, https://www.nltk.org/
10. Scikit aprende, http://scikit-learn.org/
11. Hadoop, https://hadoop.apache.org/
12. Chispa, https://spark.apache.org
13. J. Herrera, G. Molto, Detección de eventos en streaming multimedia con técnicas de big data, en *Actas de 2016 de la 24a Conferencia Internacional Euromicro sobre Procesamiento en Paralelo, Distribuido y Basado en Red (PDP),* Heraklion Creta, Grecia (2016), págs. 345–349
14. S. Hershey, S. Chaudhuri, DP Ellis, JF Gemmeke, A. Jansen, RC Moore, M. Plakal, D. Platt, RA Saurous, B. Seybold et al., Arquitecturas CNN para clasificación de audio a gran escala, en *2017 IEEE International Conference on Acustics, Speech and Signal Processing (ICASSP)* (IEEE, 2017), págs. 131-135, https://www.tableau.com/ (14. Tableau)
15. OpenCV, https://opencv.org/
16. Librosa, https://librosa.github.io/librosa/
17. K. Alex, I. Sutskever, EH Geoffrey, *Clasificación de ImageNet con Neural convolucional profundo Redes (* 2012), págs. 1097-1105
18. J. Schmidhuber, Aprendizaje profundo en redes neuronales: una descripción general. Neural Netw. **61,** 85-117 (2015)

19. C. Harris, M. Stephens, Un detector combinado de esquinas y bordes, en *Conferencia de Alvey Vision,* vol. 15, no. 50 (1988), págs. 10–5244

20. T. Lindeberg, Transformación de características invariantes de escala (2012)

21. E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: una alternativa eficiente a SIFT o SURF, en *2011 IEEE International Conference on Computer Vision (ICCV) (* IEEE, 2011), págs. 2564-2571

22. N. Dalal, B. Triggs, Histogramas de gradientes orientados para la detección humana, en *Computadora IEEE Conferencia de la Sociedad sobre Visión por Computador y Reconocimiento de Patrones, 2005, CVPR 2005,* vol. 1 (IEEE, 2005), págs. 886–893

23. Datos sobre la calidad del aire, https://archive.ics.uci.edu/ml/datasets/Air+quality