Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Telemática

Trabajo Fin de Grado

Procesamiento de algoritmos de visión artificial en la nube – Amazon AWS

Autor: Ricardo Alfonso Casanova Lozano

Tutor: Alfredo Gardel Vicente

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería Telemática

Trabajo Fin de Grado

Procesamiento de algoritmos de visión artificial en la nube – Amazon AWS

Autor: Ricardo Alfonso Casanova Lozano

Tutor: Alfredo Gardel Vicente

TRIBUNAL:

Presidente: María Soledad Escudero Hernanz.

Vocal 1º: José Antonio Jiménez Calvo.

Vocal 2º: Alfredo Gardel Vicente.

FECHA: Julio 2021

A mi familia y amigos, quienes han sido un apoyo fundamental durante toda la etapa académica.

Gracias por su cariño, valores, consejos y ayuda en todo momento.

Esto es posible gracias a ustedes.
¡¡A por más éxitos!!

Agradecimientos

Este trabajo no habría sido posible sin el apoyo y estímulo de mi tutor Alfredo, quien compartió sus conocimientos, su visión del proyecto y algunos recursos para poder culminar con éxito el trabajo. Gracias por brindarme la posibilidad de conocer más de cerca el mundo de la Inteligencia Artificial, porque es un área que me ha llamado la atención y de la que me gustaría seguir aprendiendo.

También me gustaría agradecer a todos los profesores que he tenido a lo largo de mis estudios en la Universidad, porque he aprendido de cada uno de ellos, gracias por su dedicación.

No puedo terminar sin agradecer a mis padres, Eduardo y Sandra; a mi hermano y cuñada, Luis y Ericka; a mis abuelos y demás familia por ser el motor que siempre me impulsa a buscar nuevas metas. A mis amigos, en especial, Luis y Daniela, quienes han estado presentes a lo largo de todos mis estudios pese a la distancia. A los amigos que me ha regalado la Universidad, Samuel y Cristian, gracias por su apoyo y compañía desde el primer año de la carrera.

Sinceramente, mil gracias a todos.

Resumen

El presente Trabajo Final de Grado (TFG), se centra en el estudio de los servicios de Visión Artificial en la nube, con el objetivo de comprender el funcionamiento de los mismos que hacen uso de algoritmos de Machine Learning (ML).

Se ha realizado una comparativa de los diferentes servicios de Visión Artificial, tales como: Amazon Rekognition, Azure Cognitive Service (Vision), IBM Watson Visual Recognition y Google Cloud Vision API. Para comprender de esta forma la ventaja de utilizar Rekognition frente a las otras herramientas.

Para poder hacer uso del servicio de Inteligencia Artificial de AWS (Amazon Web Services), se procedió a crear una cuenta de AWS y a solicitar créditos mediante un programa especial que tiene Amazon. Como se deseaba hacer uso de los algoritmos de Rekognition se creó una instancia virtual, la cual se configuró con la instalación de los paquetes básicos y del SDK (Software Development Kit) de Amazon Rekognition. Finalmente, para poder hacer uso de los servicios de AWS en la instancia se creó un usuario IAM (Identity and Access Management), al cual se le asignaron los permisos necesarios para que la instancia pudiese hacer uso del SDK, del bucket de Amazon S3, entre otros servicios de AWS.

Para comprender el funcionamiento del algoritmo de reconocimiento de Amazon, se han planteado varios casos de uso. El primero sirve para comprender el algoritmo propio que Amazon utiliza para detectar todos los objetos de una imagen. Rekognition también ofrece al usuario la posibilidad de crear sus propios modelos de reconocimiento, mediante el entrenamiento y uso de etiquetas personalizadas. Para este caso se realizarán una serie de pruebas, tanto en la instancia virtual creada anteriormente como en nodos Internet of Things (IoT).

Palabras claves: Visión Artificial; Aprendizaje automático; Etiquetas Personalizadas; Amazon Rekognition.

Abstract

This Bachelor's Degree Final Project focuses on the study of Artificial Vision services in the cloud, with the goal of understanding the operation of the service, wich makes use of Machine Learning (ML) algorithms.

A comparison of different Artificial Vision services has been made, such as: Amazon Rekognition, Azure Cognitive Service (Vision), IBM Watson Visual Recognition and Google Cloud Vision API, to understand in this way the advantage of using Rekognition over the other tools.

In order to use the AWS (Amazon Web Services) Artificial Intelligence service, we proceeded to create an AWS account and request credits through a special program that Amazon has. As we wanted to make use of Rekognition algorithms, a virtual instance was created, wich was configured with the installation of the basic packages and the Amazon Rekognition SDK (Software Development Kit). Finally, in order to make use of AWS services on the instance, an IAM (Identity and Access Management) user was created, who was assigned the necessary permissions so that the instance could use the SDK, the Amazon S3 bucket, among other AWS services.

To understand how Amazon's recognition algorithm works, several use cases have been proposed. The first is to understand the algorithm that Amazon uses to detect all objects in an image. Rekognition also offers the user the possibility of create their own recognition models, by training and using custom labels. For this case, a series of tests will be carried out on the previously created virtual instance and on the Internet of Things (IoT) nodes.

Keywords: **Artificial Vision; Machine Learning; Custom Labels; Amazon Rekognition.**

El tamaño de tu éxito será
del tamaño de tu esfuerzo.

Francisco de Miranda.

Índice general

Índice de figuras.

Índice de tablas.

1. Introducción.

En este primer capítulo se busca exponer de forma breve los conceptos básicos y más importantes del presente Trabajo Fin de Grado (TFG), debido a que es necesario explicar los fundamentos de la Inteligencia Artificial (IA) y la Visión Artificial, para así poder comprender el auge que está tecnología está teniendo en el mercado. De igual forma, se explicarán los dos algoritmos más usados en la IA, como lo son el Machine Learning (ML) y el Deep Learning (DL). A pesar de que ambas técnicas buscan imitar la forma en la que el cerebro humano aprende, son dos funcionamientos completamente distintos como se verá posteriormente. Dada la gran complejidad de los sistemas de entrenamiento, los proveedores de servicios en la nube se han posicionado para ofrecer aplicaciones y productos relacionados con los sistemas de IA.

Tras esta breve introducción a la temática, se procederá a marcar los objetivos que se desean lograr con este trabajo y se explicará cómo se llevarán a cabo. Estos objetivos permitirán conocer más a fondo la tecnología de reconocimiento de objetos y las ventajas de dicho uso. Para finalizar el capítulo de Introducción se procederá a exponer la estructura general del trabajo con una breve explicación de cada uno de los capítulos siguientes.

1.1. Inteligencia Artificial.

La Inteligencia Artificial, también conocida por sus siglas como IA, se puede definir como “la capacidad de las máquinas para usar algoritmos, aprender de los datos y usar el conocimiento que aprenden de ellos para tomar decisiones, al igual que lo haría un ser humano” [1].

Dentro de las principales ventajas que ofrece la IA frente al procesamiento del cerebro humano destacan dos. La primera ventaja es que para determinados problemas donde existe un gran conjunto de datos y un buen entrenamiento del sistema artificial, el margen de error en una máquina es relativamente menor con respecto a si la tarea es realizada por sus homólogos humanos. La segunda razón es que, a diferencia de los humanos, las máquinas no necesitan descansar y pueden analizar grandes cantidades de información simultáneamente y de forma permanente.

La idea de que las máquinas puedan aprender y a la vez tomar decisiones es muy importante y poderosa, es por ello que su desarrollo ha crecido de manera exponencial en los últimos años. En la actualidad se puede observar que los sistemas de inteligencia artificial realizan tareas que antes eran exclusivamente para los humanos. La ayuda de

estas tecnologías trae a las personas mejoras significativas y permite que aumente su rendimiento en todos los ámbitos de la vida.

La IA es aplicable a casi cualquier situación, por lo que su crecimiento es considerablemente rápido. Algunas aplicaciones de técnicas de inteligencia artificial son las siguientes [2]:

- Reconocimiento de imágenes, clasificación y etiquetado.
- Detección y clasificación de objetos.
- Procesamiento eficiente y escalable de datos.
- Protección contra amenazas en la ciberseguridad.
- Generar asistentes virtuales con conocimientos sobre un área compleja y específica.
- Minería de datos.
- Procesamiento del lenguaje natural.

Como se puede observar la inteligencia artificial puede generar un impacto en casi todos los sectores. También es importante resaltar que uno de los beneficios de la IA es que las máquinas pueden realizar ahora todas esas tareas que el humano pensaba que era imposible de realizar puesto a que eran complejas, aburridas o peligrosas, creando una oportunidad para el desarrollo de dichas actividades.

De igual forma es importante resaltar que la IA puede resultar en muchas situaciones incómodas para algunas personas debido a que es un tema que puede llegar a considerarse una amenaza debido a que, como ocurre en la ciencia ficción, se piensa que el desarrollo de las máquinas puede representar un peligro para la humanidad. La realidad es que se debe de ser responsable tanto con el desarrollo como el uso que se le dé a la tecnología y buscando siempre el desarrollo de tecnologías que permitan mejorar la vida de las personas.

La característica principal de la Inteligencia Artificial es que no hay que programarla específicamente para cada escenario, sino que se le puede enseñar cosas a la máquina que le permita realizar la predicción (Machine Learning) o que la máquina pueda aprenderlo por si misma (Deep Learning). En la **Figura 1-1** se puede observar una comparación de la IA, el ML y el DL:

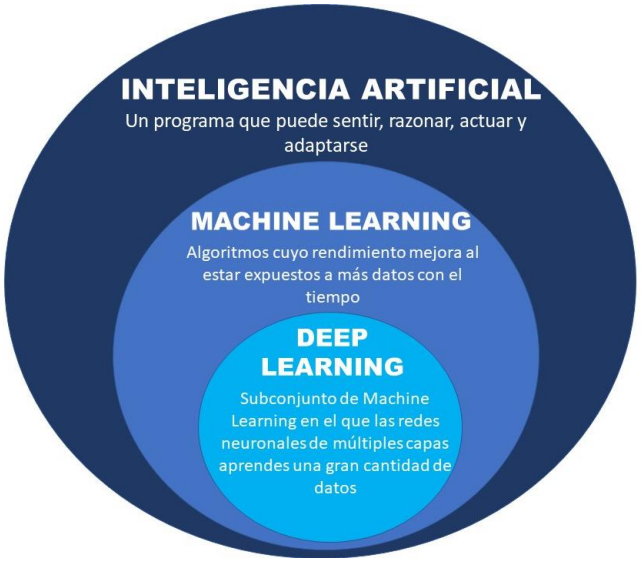


Figura 1-1: Comparación IA, ML y DL. *[3]*.

1.2. Machine Learning.

Se puede definir el Machine Learning (ML), como una rama de la Inteligencia Artificial, la cual, mediante el uso de algoritmos, dota a las computadoras de la capacidad de identificar patrones en datos masivos que le permitan realizar predicciones [4].

Esta tecnología necesita proporcionar datos estructurados y clasificados al sistema, para que pueda deducir como clasificar los nuevos datos. Dependiendo de la clasificación, el sistema realizará la acción que tiene programada.

Lo que se busca con el ML, es que las máquinas aprendan imitando la forma analítica que tienen los humanos de aprender, mediante la identificación, clasificación o predicción. Esto lo realiza adaptando el algoritmo después de recibir el feedback de una persona, realimentando de esta forma el sistema con datos estructurados y categorizados.

Se puede decir que la idea del Machine Learning fue planteada en el año 1950 por el matemático Alan Turing, cuando planteó la posibilidad de que las máquinas pudiesen "pensar" [5].

Esta tecnología se aplica en plataformas digitales como Netflix o Spotify, en las respuestas inteligentes de Gmail, en los asistentes inteligentes como Siri o Alexa, en la robótica, en coches autónomos, en el diagnóstico médico, etc.

En la **Figura 1-2** se muestra un ejemplo habitual de los pasos y secciones de un algoritmo de ML. En primer lugar, se tienen los datos de entrada, por ejemplo, una imagen

donde se tiene capturado un coche. A partir de esos datos se obtienen múltiples características que se consideran de interés o relevancia para que en la etapa posterior se aplique un algoritmo de clasificación entrenado previamente y que obtenga en la salida la detección o no de la existencia de un coche en la imagen de entrada.

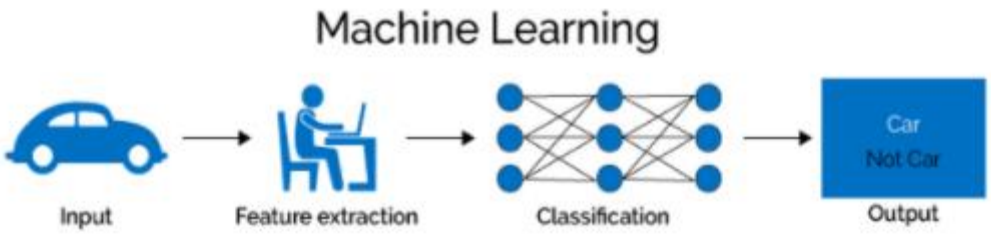


Figura 1-2: Funcionamiento del Machine Learning. *[6]*.

1.3. Deep Learning.

Por su parte, Deep Learning (DL) es un algoritmo automático jerárquico que busca simular el aprendizaje humano con la finalidad de obtener ciertos conocimientos. La particularidad de este algoritmo es que no necesita de reglas programadas con anterioridad, sino que el mismo sistema tiene la capacidad de aprender por sí mismo, gracias a una fase previa de entrenamiento [7]. La gran diferencia con ML es que el sistema no tiene de partida conocimiento de qué características son relevantes para la clasificación o detección de patrones que se desea realizar. De esta manera, cualquier información oculta a primera vista y que pueda ser utilizada en la clasificación posterior, será tenida en cuenta.

El Deep Learning también se caracteriza por usar redes neuronales artificiales entrelazadas que permiten el procesamiento de la información, imitando de esta forma el comportamiento del cerebro. Al combinar diferentes algoritmos, el sistema puede procesar datos no estructurados. El DL se utiliza principalmente para la automatización del análisis predictivo.

Es realmente útil para las tareas complejas en donde no todos los aspectos de los objetos pueden clasificarse de antemano, por lo que el propio sistema es el que busca diferenciadores adecuados, analizando así las nuevas entradas en búsqueda de otras características que el sistema pueda utilizar para clasificar la entrada.

El aprendizaje profundo necesita una cantidad de datos más alta que la que se puede necesitar para el aprendizaje automático. Por lo que se puede deducir que el Deep Learning es una tecnología mucho más compleja de implementar, ya que requiere de un mayor número de recursos informáticos.

El Deep Learning cuenta con 3 capas: La primera es la capa de entrada (neuronas que asimilan los datos de entrada), la segunda es la capa oculta (red que realiza el procesamiento de la información y cálculos intermedios) y la capa de salida (es donde se toma la decisión).

El aprendizaje profundo se puede aplicar en: traductores inteligentes, en sistemas de reconocimiento de voz, en lenguaje natural hablado y escrito, en sistemas de reconocimiento de imágenes, en visión computacional, etc.

En la **Figura 1-3** se observa un ejemplo habitual del funcionamiento del algoritmo de DL. En primer lugar, se encuentran los datos de entrada, por ejemplo, una imagen donde se tiene capturado un coche. Seguidamente, se procesa dicha información para realizar los cálculos intermedios que permitan al algoritmo aprender de los datos no estructurados y de esta forma generar en la salida la detección o la no existencia de un coche en la imagen de entrada.

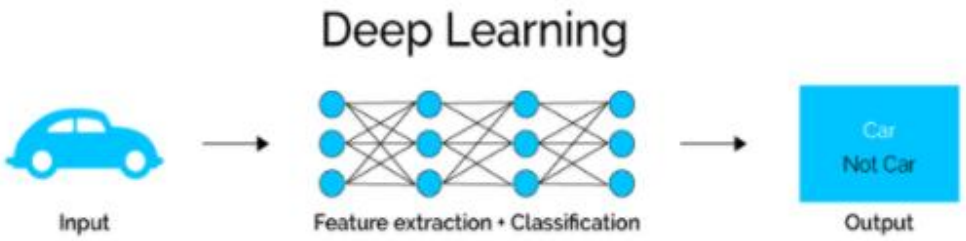


Figura 1-3: Funcionamiento del Deep Learning. *[6]*.

1.4. Visión Artificial.

Un sistema de procesamiento basado en visión artificial es un método automatizado e inteligente que puede procesar y analizar imágenes del mundo real con el fin de procesar información que la máquina pueda tratar [8]. Esto se puede realizar mediante el uso de diferentes paquetes de softwares instalados en el sistema y en las cámaras, de manera que las computadoras pueden llegar a detectar o identificar productos, objetos, eventos, acciones, etc.

La visión artificial es una herramienta importante para verificar y detectar cualquier error en un proceso de producción, por lo que ayuda a resolver tareas industriales complejas de forma confiable y consistente.

La finalidad de la visión artificial consiste en proporcionar a las máquinas un sistema de reconocimiento visual para conocer lo que está sucediendo en el mundo real, de modo que puedan tomar decisiones que automaticen cualquier proceso.

Algunos de los usos más característicos de visión artificial son: reconocimiento de objetos, control de calidad, posicionamiento, control de rotación, contaje de productos, clasificación, etc.

Actualmente los sistemas de visión artificial están introduciendo algoritmos de Machine Learning y Deep Learning, que logran superar los niveles de detección, reconocimiento y clasificación de técnicas tradicionales.

1.5. Servicios de IA en la nube

Los servicios de Inteligencia Artificial en la nube son las herramientas de aprendizaje automático, diseñadas y previamente entrenadas por las grandes compañías de tecnología como Amazon, Google y Microsoft, que facilitan la implementación y uso de los sistemas de la IA, permitiendo que sea utilizado por un público mucho más amplio y que no necesita tener experiencia en los algoritmos de Machine Learning.

Este modelo de negocio les permite a otras compañías hacer uso de estas tecnologías a un costo más accesible. Debido a la facilidad que tienen los servicios de IA en la nube de integrarse con las aplicaciones, sus casos de uso son cada vez más, que pueden ir desde robots que simulan una conversación de atención al cliente hasta algoritmos que puedan detectar tipos de cáncer con mayor precisión que los médicos mediante el análisis de imágenes de tomografías. [9]

Dentro de los servicios de Inteligencia Artificial encontramos los siguientes:

- Análisis avanzado de texto: Mediante el procesamiento de lenguaje natural se extrae información y relaciones de un texto no estructurado.

- Análisis de imagen y video: Permite agregar esta funcionalidad a las aplicaciones para catalogar recursos, automatizar flujo de trabajo o extraer significado.

- Traducción en tiempo real: Permite una traducción eficaz y rentable a varios idiomas.

- Texto a voz o transcripción de voz a texto.

Como se ha mencionado anteriormente existen múltiples implementaciones para los servicios de Inteligencia Artificial, ya que muchas veces depende del HW (Hardware) que se vaya a utilizar. El crecimiento de este tipo de servicios se debe a que permiten entrenar los sistemas de IA mediante el uso de grandes cantidades de datos y porque

proporcionan una gran potencia de cálculo. El uso de la nube permite que la ejecución de los servicios de IA en la red no necesite de tantos recursos informáticos.

1.6. Nodos IoT/Edge.

Para empezar, es necesario explicar que el IoT (Internet of Things) se entiende como una red de dispositivos conectados que permiten monitorizar, compilar y analizar datos con la finalidad de tomar una decisión. Por lo tanto, los nodos IoT son los elementos físicos dentro de un entorno IoT, que permiten la conexión del mundo físico con Internet [10], dentro de estos dispositivos se encuentran los sensores, máquinas y localizadores GPS.

Los nodos IoT deben de cumplir con una de estas dos funciones:

- **Recoger datos y enviarlos** a Internet mediante una interfaz de comunicación.
- **Actuar** sobre el medio en base a los datos recibidos de Internet.

De igual forma, los nodos IoT pueden tener la capacidad de procesamiento local en el propio dispositivo, por lo que en este caso pasa a considerarse como un nodo Edge, debido a que permiten acercar el almacenamiento y el procesamiento de los datos a la ubicación desde la que se utilizarán, permitiendo agilizar el proceso. El uso de estos nodos permite ahorrar en costes, abrir nuevas vías de procesos o mejorar la calidad del servicio.

1.7. Objetivos.

El objetivo de este proyecto es realizar un análisis donde se expliquen los pasos a seguir para saber cómo utilizar un servicio de IA en la nube como es Amazon Rekognition. Se desea, por un lado, detallar la operativa del sistema en la nube y de igual forma, poner en marcha un ejemplo que permita comprender el funcionamiento de los algoritmos de procesamiento de visión artificial.

Dado que existen diferentes sistemas de IA en la nube, el TFG también busca conocer y comparar las ventajas e inconvenientes de usar Amazon Rekognition frente a otras soluciones como, por ejemplo, Cognitive Service de Microsoft Azure.

De igual forma, se quiere realizar un estudio del funcionamiento de las etiquetas personalizadas de Rekognition y llegar a implementar la detección de objetos de forma remota.

Para alcanzar estos objetivos se ha seguido el plan de trabajo mostrado a continuación:

- **Documentación y estudio de Amazon Recognition:** En esta primera fase se ha buscado información para tener los conocimientos básicos que son necesarios para entender el estado actual del arte y de las tecnologías similares que se pueden usar.

- **Planteamiento de los casos de uso existentes:** Se desea revisar sobre buenas prácticas y casos de éxito de Rekognition en empresas reconocidas y en proyectos que resultan interesantes para comprender el alcance del servicio.

- **Desarrollo de un caso de uso:** En el TFG se desarrolla un caso de uso, donde se pueda comprender el uso de las etiquetas personalizadas, creando el modelo de reconocimiento que se utilizará para entrenar los datos y se realizaron las pruebas necesarias, gracias al uso de un dataset con diversas imágenes que permitió realizar el entrenamiento del modelo.

- **Evaluar el funcionamiento:** Se desea evaluar que el funcionamiento del modelo es el correcto, si el porcentaje de acierto que se obtiene es inferior al deseado, se debe de aportar más imágenes al modelo para que de esta forma se pueda mejorar los porcentajes de precisión y recuperación.

1.8. Estructura del TFG.

En este apartado se especifica la estructura básica de la memoria del TFG, incluyendo un breve resumen con los aspectos más relevantes y significativos de cada capítulo.

Capítulo 1: Introducción. Contiene una introducción sobre la información general que se debe de conocer para abordar el trabajo de la mejor forma. De igual manera, se explican los objetivos del TFG y como se plantea alcanzarlos.

Capítulo 2: Estado del Arte. Se documentan las características principales de Amazon Rekognition, además se estudian las otras herramientas que prestan el servicio de reconocimiento de objetos (Google, Azure e IBM). Para finalizar se realiza una comparativa de los servicios de visión artificial.

Capítulo 3: Diseño del servicio. Se desarrollan los pasos necesarios para crear y configurar una máquina virtual, permitiendo de esta forma el desarrollo en el siguiente capítulo del reconocimiento de objetos.

Capítulo 4: Desarrollo del reconocimiento de objetos. Se describen los pasos necesarios para realizar el reconocimiento de objetos en una imagen, también se indica como preparar un modelo de entrenamiento, de forma que permita reconocer objetos mediante etiquetas personalizadas.

Capítulo 5: Reconocimiento de objetos desde un nodo IoT/Edge: En este capítulo se indica el procedimiento para aplicar en una Raspberry Pi el SDK de Amazon Rekognition, como realizar el reconocimiento de etiquetas personalizadas en una Raspberry. Para finalizar, se integra la placa con el módulo de cámara Raspberry Pi, logrando de esta forma reconocer objetos en imágenes capturadas en tiempo real.

Capítulo 6: Conclusiones y trabajo futuro. Se describen las conclusiones del trabajo realizado y se presenta una vía de trabajo futuro para este proyecto.

Bibliografía y referencias. Se añaden los libros, artículos y materiales que han sido consultados para realizar la elaboración de este TFG. Se ha aplicado el estilo de citación recomendado por la Universidad de Alcalá (UAH).

Anexos. Se incluyen los códigos de los diferentes programas que se han desarrollado para poder aplicar los algoritmos de detección y los que permiten hacer uso del SDK de Rekognition para activar o desactivar los modelos de reconocimiento de objetos. Para finalizar, se incluyen diferentes casos de uso de Amazon Rekognition

2. Estado del arte.

En este capítulo se explica en detalle la plataforma Amazon Rekognition, comprendiendo el funcionamiento del servicio y las diversas opciones de reconocimiento que esta herramienta posee. Se aportan varios casos en los que se ha usado el servicio de Rekognition y se comentan las diversas aplicaciones que se le puede dar al reconocimiento de objetos en una imagen. También se compara el servicio de reconocimiento de otras empresas como MS Azure, Google Cloud Vision e IBM Watson con la herramienta Amazon Rekognition, demostrando de esta forma las ventajas que cada una aporta.

2.1. Amazon Rekognition.

Rekognition es la herramienta de reconocimiento facial, análisis de imágenes y videos de Amazon, por lo que puede identificar objetos, personas, texto, escenas, emociones, actividades e incluso detectar contenido inadecuado [11], tal como se muestra en la **Figura 2-1**. Es una API (Application Programming Interface) simple y fácil de usar, puede analizar cualquier imagen o video, los archivos pueden ser proporcionados por el usuario de forma local o se pueden encontrar en el servicio de almacenamiento de objetos conocido como Amazon S3 (Amazon Simple Storage Service).

Amazon Rekognition siempre está en constante aprendizaje, gracias a los nuevos datos que aportan los usuarios, convirtiéndola cada vez en una herramienta más inteligente. Una de las ventajas de usar el servicio de Amazon es que el usuario tiene la opción de configurar la etiqueta que desee, pudiendo identificar objetos o escenas personalizadas.

Figura 2-1: Amazon Rekognition. *[11]*

2.1.1. Características principales de Amazon Rekognition.

En este punto, se explican las características de Rekognition, logrando de esta forma conocer cómo funciona el reconocimiento de imágenes y videos, y comprendiendo las diferentes opciones de detección que la plataforma ofrece.

Antes de iniciar, es necesario explicar el significado que le da Amazon Rekognition a las etiquetas, se pueden definir como la escena, actividad, objeto que el algoritmo de visión artificial programado por Amazon logra detectar [11]. Como se menciona más adelante, dichas etiquetas se pueden personalizar, logrando de esta forma que el usuario pueda detectar en una imagen o video el objeto que estime necesario.

Por lo tanto, Rekognition sirve para detectar las diferentes etiquetas que detecte en imágenes y videos, es por ello que cuenta con dos servicios como lo son: Amazon Rekognition Image y Amazon Rekognition Video.

Amazon Rekognition Image, es un servicio de reconocimiento de imágenes que utiliza la tecnología de aprendizaje profundo (Deep Learning), que permite la detección de objetos, escenas y rostros, así como otras actividades que veremos más adelante. Rekognition Image se basa en la misma tecnología de Deep Learning probada y altamente escalable, desarrollada por los científicos de visión artificial de Amazon, la cual les permite analizar millones de imágenes al día para el servicio Prime Photos. [12]

Dicho servicio retorna una puntuación de fiabilidad de los elementos que reconoce para que el usuario pueda tomar una decisión en base a los resultados. También devuelve las coordenadas de una caja rectangular, la cual define el objeto reconocido, facilitando de esta manera la localización del objeto en la imagen.

- Detección de objetos y escenas: Se identifican miles de objetos y escenas dentro de una imagen (como la playa o la puesta del sol). Facilitando la operación de buscar, filtrar y preparar bibliotecas de imágenes de gran tamaño.
- Reconocimiento facial: Permite reconocer rostros de una gran colección de imágenes, proporcionando las caras que más se parecen al rostro de referencia.
- Análisis facial: Localiza rostros en imágenes y las analiza reconociendo si la persona está sonriendo, tiene los ojos abiertos, etc.
- Comparación de rostros: Permite establecer las probabilidades de que las caras en dos imágenes diferentes sean de la misma persona.
- Detección de imágenes no seguras: Detecta imágenes con contenido explícito o sugerente, permitiendo filtrar en función de los requisitos del usuario.

- Reconocimiento de famosos: Detecta y reconoce a personajes famosos, destacados o importantes en sus ámbitos, lo que permite reconocer en imágenes personas famosas según las necesidades de marketing.
- Texto en imágenes: Localiza y extrae el texto de imágenes, tales como carteles, texto sobre objetos, pantallas o noticias. Devolviendo una etiqueta con el texto detectado.
- Detección de equipo de protección personal (EPP): Permite detectar si una persona está usando el equipo de protección, y si este cubre la parte del cuerpo correspondiente.

Amazon Rekognition Video, es un servicio de análisis de video que aplica la tecnología de aprendizaje automático (Machine Learning), que permite la detección de objetos, escenas, celebridades, actividades y cualquier tipo de contenido no adecuado. Del mismo modo, proporciona servicios precisos de análisis de rostros con funciones de búsqueda de rostros para detectarlos, analizarlos y compararlos [13].

Los resultados se matchean con una marca temporal, creando de esta forma un índice para búsquedas detalladas del video o especificar la parte del video en la que se quiere realizar un análisis más profundo. También devuelve las cajas rectangulares con el reconocimiento de objetos, rostros, textos y personas.

Una de las ventajas de este servicio es que puede monitorear una transmisión en tiempo real, haciendo uso de Amazon Kinesis Video Streams, lo que le permite detectar y buscar rostros a partir de los datos proporcionados por el usuario, puede incorporar subtítulos o filtrar escenas obscenas y transcribir el video complementándose con el uso de Amazon Transcribe.

- Detección de actividades, objetos y escenas: Identifica miles de objetos, escenas y actividades que puede estar realizando una persona.
- Moderación de contenido: Detecta contenido inapropiado y proporciona marcas temporales para cada detección.
- Detección de texto: Detecta y lee texto de los videos, permitiendo el filtrado de palabras por regiones de interés.
- Reconocimiento de celebridades: Detecta cuando y donde aparecen personas famosas en el video.
- Detección y análisis de rostros: Puede detectar hasta 100 caras en un cuadro de video y de igual forma, se pueden obtener atributos de la persona.
- Análisis de video en directo: Permite analizar videos en tiempo real para buscar y detectar rostros.

2.1.2. Clientes conocidos de Amazon Rekognition.

El servicio de Amazon Rekognition puede hacer que la detección en la imágenes y videos sean útiles para cualquier empresa, porque se puede implementar desde la clasificación de partes específicas de una máquina en la línea de ensamblado hasta llegar a detectar problemas en una planta. Dentro de los clientes que usan Rekognition se pueden encontrar muchas empresas reconocidas, tales como [11]:

- CBS (empresa de medios de comunicación): Usan Rekognition para moderar el contenido inapropiado, de forma que puedan cumplir las regulaciones en cada país. Realizando la detección y edición en casi tiempo real, mientras que hacen uso de las etiquetas personalizadas para mejorar los modelos de moderación.
- NFL (National Football League): Usan las etiquetas personalizadas para generar automáticamente etiquetas de metadatos adaptadas al negocio y proporcionar facetas de búsqueda para el equipo de producción.
- THORN: Es una organización sin fines de lucro dedicada a detener la propagación de material de abuso infantil. Han creado con Rekognition herramientas para encontrar a los niños más rápido y poner fin a la programación de material de abuso sexual infantil.
- K-STAR Group: Es una empresa de entretenimiento, crearon con Rekognition el servicio "Face Ticket" donde los asistentes a los conciertos podrán verificar la compra del boleto y acceder al concierto sin hacer filas.
- Artfinder: Mercado de arte en línea que permite a los artistas vender directamente a los compradores. Utilizan una herramienta de recomendación que permite unir a los clientes con el arte que les encantará.
- FamilySearch: Organización genealógica que se dedica a conectar familias a través de generaciones. Usan Rekognition para ayudar a los usuarios a identificar a cuál de sus antepasados se parecen más, gracias al análisis de las fotos familiares.

2.2. Azure Cognitive Service: Vision.

Cognitive Service es el producto de Microsoft Azure que permite a los desarrolladores hacer uso de la inteligencia artificial sin necesidad de tener conocimientos en Machine Learning, por lo que con una simple llamada a la API se puede agregar características de IA a las aplicaciones [14]. Dentro de los servicios que ofrece Cognitive encontramos:

- Decisión: Se utiliza para realizar la toma de decisiones de una forma mucho más inteligente y ahorrando en tiempo. Permite realizar una detección temprana de los posibles problemas, detectar contenido que

puede resultar ofensivo o no deseado y crear experiencias personalizadas muy completas.

- Lenguaje: Permite extraer significado a partir de texto sin estructura. Dentro de las aportaciones encontramos que se puede incorporar reconocimiento del lenguaje natural a las aplicaciones, bots o dispositivos IoT (Internet of Things); otra opción es la de crear una capa conversacional de preguntas y respuestas basada en los datos, también ofrece los servicios para detectar opiniones, frases clave, entidades con nombre o traducir hasta en más de 90 idiomas.
- Voz: Este servicio permite integrar los servicios de voz en las aplicaciones. Permite transcribir la voz a texto legible o convertir el texto en lenguaje más real que permita obtener interfaces más naturales, integrar la traducción en tiempo real a las aplicaciones o identificar qué persona está hablando, basándose en audio (este servicio está en versión preliminar).
- Visión: Permite identificar y analizar el contenido en imágenes y vídeos, por lo tanto, este servicio de IA de Microsoft es el homólogo al de Amazon Rekognition. Vision también permite personalizar el reconocimiento de la imagen para adaptarlo a las necesidades empresariales y permite detectar e identificar a las personas y emoticonos en las imágenes. Se puede ejecutar tanto en la nube como en contenedores y su implementación permite optimizar los procesos.

Todos ellos integran los diferentes servicios cognitivos ofrecidos por Microsoft Azure, mostrando la red neuronal que ha creado MS, tal y como se observa en la **Figura 2-2**:

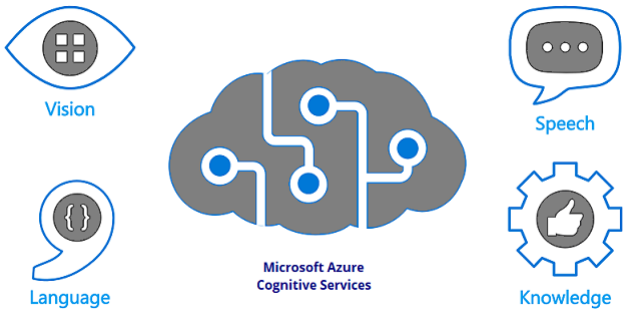


Figura 2-2: Microsoft Azure Cognitive Services. *[15]*

El servicio de Vision de Azure se divide en 3 productos: Computer Vision, Custom Vision y Face. El primer producto ofrece la posibilidad de procesar los datos visuales, de forma que se pueda etiquetar el contenido de objetos y conceptos, automatizar la extracción de texto, generar descripciones de una imagen, moderar su contenido y reconocer el movimiento de personas en videos.

Por su parte, Custom Vision permite crear un modelo de visión artificial personalizado en solo unos minutos, para ello se puede entrenar el modelo simplemente cargando y etiquetando las imágenes. Luego el modelo realiza por su cuenta las pruebas y mejora la precisión de forma constante conforme se agregan imágenes nuevas. Azure cuenta con una interfaz que permite guiar al usuario en el desarrollo y en la implementación de los modelos de visión artificial personalizados.

Para finalizar encontramos el producto Face, el cual es el servicio de inteligencia artificial que analiza las caras en una imagen. Dentro de las características que incluye se encuentra: detección de rostros y de los atributos, identificación de la persona que coincide con el individuo (cuenta con un repositorio privado de hasta 1 millón de personas), reconocimiento de las emociones en base a las expresiones faciales y, por último, detección y agrupación de caras similares.

2.3. IBM Watson Visual Recognition.

Este servicio ofrecido por IBM utiliza algoritmos de Deep Learning para analizar imágenes de escenas, objetos y otros contenidos. Devolviendo como respuesta palabras clave que proporcionan información del contenido detectado. Visual Recognition cuenta con varios modelos que no necesitan de entrenamiento para realizar la detección, dentro de los cuales encontramos los siguientes [16]:

- Modelo General: Cuenta con miles de etiquetas, por lo que devuelve clases organizadas en categorías y subcategorías.
- Modelo Explícito: Este modelo permite identificar las imágenes inadecuadas.
- Modelo de alimentos: Por su parte, este es un modelo específico para identificar imágenes de alimentos.
- Modelos Personalizados: Estos modelos si son necesarios entrenar porque permiten crear clases especializadas.

Los tres primeros modelos funcionan de la siguiente forma: se preparan las imágenes que se desean analizar, se estudian las imágenes en base a las características incorporadas y se obtienen los resultados. En el caso de que sea necesario entrenar el modelo, la metodología es la siguiente: se preparan los datos de entrenamiento (incluyendo imágenes positivas o negativas), se pasa a entrenar y crear los nuevos modelos con los datos de entrada, luego se analizan las imágenes con el modelo personalizado y finalmente se obtienen los resultados. Esta descripción se puede detallar mejor en la **Figura 2-3**:

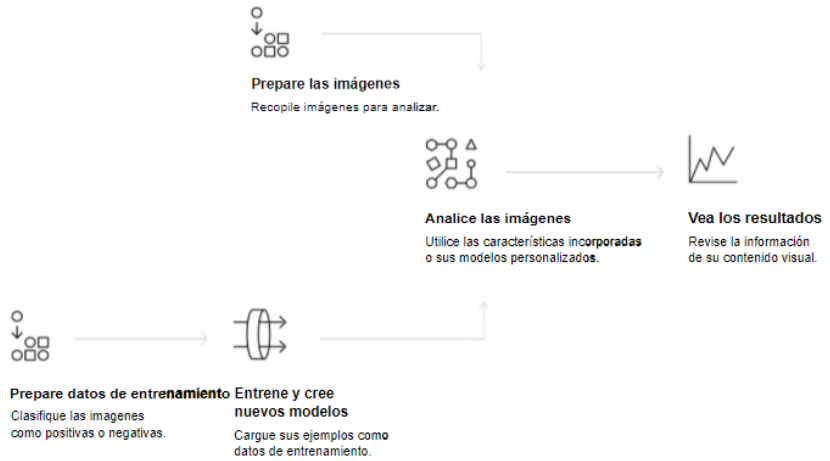


Figura 2-3: Funcionamiento de IBM Watson Visual Recognition. *[16]*

2.4. Google Cloud Vision API.

Vision AI, de Google Cloud, permite extraer información de las imágenes en la nube o en el perímetro, mediante AutoML Vision o de igual forma se puede hacer uso de modelos previamente entrenados de la API que permitan detectar emociones, interpretar texto, etc. Los dos productos de visión artificial de Google Cloud utilizan Machine Learning al momento de realizar la predicción de las imágenes.

El producto AutoML Vision permite automatizar el entrenamiento de los modelos personalizados, únicamente se necesitan subir los datos de entrenamiento y se entrenará el modelo mediante la interfaz gráfica, facilitando la optimización del modelo, la latencia, el tamaño y permitiendo exportarlo a una aplicación en la nube o a dispositivos en el perímetro [17]. De esta forma se pueden ejecutar los modelos en los dispositivos IoT, gracias a la aplicación conjunta del modelo AutoML con el servicio Edge Connect, tal y como se muestra en la **Figura 2-4**.

Figura 2-4: Funcionamiento de Google Vision AI en dispositivos IoT. *[17]*

Por su parte, la API (Application Programming Interface) de Vision ofrece modelos de aprendizaje automático previamente entrenados y potentes, mediante las APIs REST y RCP (Remote Procedure Call). Al analizar las imágenes, les asigna una etiqueta y permite clasificarla dentro de millones de categorías predefinidas. Detecta caras, objetos, interpreta texto y consigue metadatos que pueden resultar útiles para el catálogo de imágenes.

2.5. Comparativa de los servicios de Visión Artificial.

En este punto se busca obtener mediante las páginas web oficiales de los 4 servicios, toda la información que nos permita conseguir el detalle de las características que brinda cada uno de los servicios y comparar entre sí las herramientas de reconocimiento de imágenes.

Para poder realizar la comparación de forma correcta, se hará uso de la **Tabla 2-1,** la cual permite resaltar las características más importantes de cada una de las herramientas.

TABLA COMPARATIVA DE HERRAMIENTAS DE VISIÓN ARTIFICIAL.				
Características	**Amazon Rekognition**	**Azure Computer Vision**	**Watson Visual Recognition**	**Google Cloud Vision API.**
Empresa desarrolladora	Amazon.	Microsoft Azure.	IBM.	Google.
Plataformas Soportadas	SaaS.	Windows. SaaS.	Windows. Mac. SaaS.	SaaS.
Audiencia del servicio	Organizaciones para automatizar el análisis de imágenes y video con ML.	Equipos de ML.	Equipos de ML.	Equipos de ML.
Soporte del servicio.	Online.	Horas de trabajo. Online	Horas de trabajo. Online	Horas de trabajo. Soporte 24/7. Online
Precio.	Versión Gratis.	Prueba Gratis.	Prueba Gratis.	Prueba Gratis
Capacitación.	Documentación.	Documentación. Webinars. Soporte Online.	Documentación. Webinars. Soporte Online.	Documentación. Webinars. Soporte Online.
Etiquetas personalizadas.	Si.	Si.	Si.	Si.
Integración.	Amazon A2I. Amazon Web Services (AWS). BotCore.	Microsoft Azure.	BotCore. IBM Cloud.	BotCore. Google Cloud Dataflow.

Tabla 2-1: Tabla comparativa de herramientas de Visión Artificial *[18]***.**

A continuación, se observa en la **Tabla 2-2** las diferentes detecciones que se realizan en cada una de las herramientas como: el reconocimiento facial, de objetos, celebridades, el análisis de imágenes y de vídeos, entre otros.

	Google	Azure	Rekognition	IBM
Detección de rostros	SI	SI	SI	SI
Reconocimiento de rostros	NO	SI	SI	NO
Puntos de referencia faciales	SI	SI	SI	NO
Detección de características	SI	SI	SI	NO
Caras similares	NO	SI	SI	NO
Emociones	SI	SI	SI	NO
Detección de etiquetas	SI	SI	SI	SI
Puntos de referencias	SI	SI	NO	SI
Famosos	NO	SI	SI	NO
Detección de logo	SI	NO	NO	NO
Reconocimiento óptimo de caracteres (OCR)	SI	SI	SI	SI
Detección de contenido explícito	SI	SI	SI	SI
Análisis de imagen	SI	SI	SI	SI
Análisis de video	SI	SI	SI	NO
Creación de modelos personalizados	NO	SI	NO	SI

Tabla 2-2: Comparación de las detecciones. *[19]*.

Como se detalla en la **Tabla 2-2**, 3 de las 4 herramientas (Google, Azure y Amazon) son bastante potentes en cuanto a la detección de imágenes y vídeos, mientras que IBM es un servicio que no se encuentra tan desarrollado como los otros.

Según el estudio de precisión de reconocimiento de imágenes realizado por **Perfficient Digital**, se puede observar que en base a las figuras que se han analizado, Google, Amazon y Azure son los servicios que han realizado la detección con una precisión mayor al 90% de confianza. [20]

Siendo Google la que más confianza daba con un 81.7% de precisión en las imágenes analizadas, seguida de Amazon con un 77.77%, luego se encuentra Azure con un 75.8% y de último IBM con un 55.6%, como se puede visualizar en la **Tabla 2-3**.

Puntaje general por motor de reconocimiento.			
	Preciso	**Incorrecto**	**No está seguro**
AWS Rekognition	77.7%	21.7%	0.6%
Google Vision	81.7%	16.7%	1.6%
IBM Watson	55.6%	41.7%	2.7%
Microsoft Azure	75.8%	235%	0.7%

Tabla 2-3: Estudio Realizado por Perfficient Digital. *[20]***.**

Finalmente se ha optado por escoger Amazon Rekognition como herramienta de reconocimiento, debido a que, según la comparativa realizada, Amazon se especializa en las etiquetas personalizadas, lo que permite a las organizaciones y particulares realizar una búsqueda automatizada del análisis mediante Machine Learning.

De igual forma, la fortaleza de Amazon es el dominio que tiene en el mercado de las nubes públicas y que cuenta con la red más completa de centros de datos mundiales. Para finalizar, es importante resaltar su facilidad de integrar el servicio de visión artificial con el de almacenamiento de datos llamado Amazon S3, lo que permite ahorrar costos de ancho de banda saliente y facilita el envío de archivos mucho más grandes (15Mb).

3. Diseño del servicio de IA en AWS.

En este capítulo se indica la metodología a seguir para poder realizar la configuración necesaria, que permita hacer uso del servicio de reconocimiento de objetos de Amazon. Para lograr este objetivo se explica en detalle en cada apartado como crear: una cuenta de AWS (Amazon Web Services), un usuario en el servicio IAM (Identity and Access Management) y una instancia virtual con su respectiva conexión por SSH (Secure Shell), instalarle los paquetes básicos de Ubuntu y el SDK (Software Development Kits).

3.1. Creación de la cuenta de AWS y créditos.

En este punto se procede a crear una cuenta en el servicio Amazon Web Services, también conocido como AWS, para ello se ha procedido a realizar el registro haciendo uso del correo oficial de la Universidad de Alcalá (UAH). De igual forma, para poder contar con créditos en la plataforma que permitiese poner a prueba los servicios de AWS, he aplicado al **Programa de Prueba de Concepto de AWS**, el cual ofrece 300$ de créditos, que permiten hacer uso de los servicios que no están incluidos en la capa gratuita de Amazon.

3.2. Creación de una Instancia Virtual.

Para poder hacer una primera prueba del servicio de reconocimiento de objetos, se considera necesario para el trabajo utilizar una instancia virtual que funcione como servidor y que permita realizar las configuraciones y tareas necesarias para poder hacer uso de los algoritmos de reconocimiento.

Por lo tanto, se procede a crear una instancia o también llamada máquina virtual, mediante el servicio de EC2 (Elastic Compute Cloud) de AWS. Para la configuración, es necesario elegir una imagen de Amazon Machine, la cual permite seleccionar el sistema operativo que tendrá la instancia. En este caso se ha decidido utilizar la imagen de **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type**. Como se indica en la **Figura 3-1**, debido a que es la instancia que he utilizado durante todo el estudio universitario y porque dicha imagen pertenece a la capa gratuita de AWS, por lo que Amazon brinda 750 horas gratuitas para poder hacer uso de la instancia creada.

Dentro de las demás imágenes incluidas en la capa gratuita se encuentran: Amazon Linux 2 AMI, Red Hat Enterprise Linux 8, Microsoft Windows Server, entre otras opciones adicionales a las AMI (Imágenes de Máquina de Amazon) de pago.

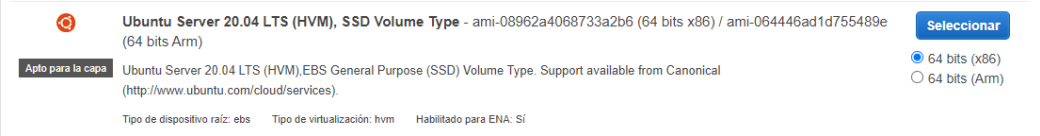
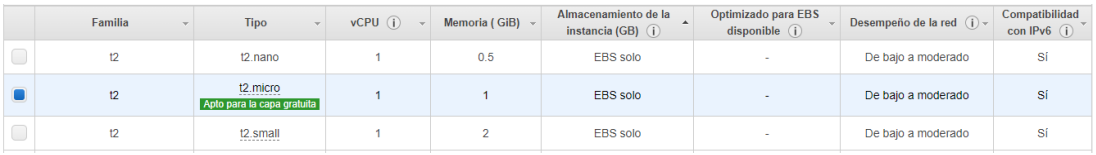


Figura 3-1: Imagen Ubuntu Server.

A continuación, se debe de seleccionar el tipo de instancia a usar, para ello se ha seleccionado la opción **t2 micro**, porque es la única instancia apta para la capa gratuita. Esta máquina virtual cuenta con 1 vCPU, 1 GB (Gigabyte) de memoria, es compatible con las direcciones de IPv6 y cuenta con un almacenamiento de 8 GB. Tal y como se pueden observar en las **Figuras 3-2** y **3-3**.



	Familia	Tipo	vCPU	Memoria (GiB)	Almacenamiento de la instancia (GB)	Optimizado para EBS disponible	Desempeño de la red	Compatibilidad con IPv6
☐	t2	t2.nano	1	0.5	EBS solo	-	De bajo a moderado	Sí
☑	t2	t2.micro Apto para la capa gratuita	1	1	EBS solo	-	De bajo a moderado	Sí
☐	t2	t2.small	1	2	EBS solo	-	De bajo a moderado	Sí

Figura 3-2: Instancia t2 micro.

Tipo de volumen	Dispositivo	Snapshot	Tamaño (GiB)	Tipo de volumen	IOPS	Velocidad (MB/s)	Eliminar al terminar	Cifrado
Raíz	/dev/sda1	snap-08d55512ce962b5e5	8	SSD de uso gener	100/3000	N/D	☑	No cifrad

Figura 3-3: Almacenamiento de la instancia.

La instancia seleccionada cumple con los requisitos mínimos necesarios para la realización de este trabajo, sin embargo, dependiendo del almacenamiento que se requieran, del número de CPUs (Central Processing Unit) virtuales, se podrá seleccionar otra de las opciones que se encuentran disponibles en la plataforma.

Seguidamente, se ha creado un grupo de seguridad que permita el acceso por SSH (puerto 22), como se visualiza en la **Figura 3-4**.

Tipo de volumen	Dispositivo	Snapshot	Tamaño (GiB)	Tipo de volumen	IOPS	Velocidad (MB/s)	Eliminar al terminar	Cifrado
Raíz	/dev/sda1	snap-08d55512ce962b5e5	8	SSD de uso gener	100/3000	N/D	☑	No cifrad

Figura 3-4: Grupo de seguridad.

Para finalizar la configuración correctamente, se debe de seleccionar o crear un nuevo par de claves, las cuales permitirá conectarse a la instancia que se ha creado de forma segura mediante SSH. Se ha creado entonces un nuevo par de claves, escribiendo

el nombre con el que se desee identificar el fichero y se procede a realizar la descarga del mismo. Una vez se haya guardado el archivo en un lugar seguro y accesible se procederá a lanzar la instancia creada.

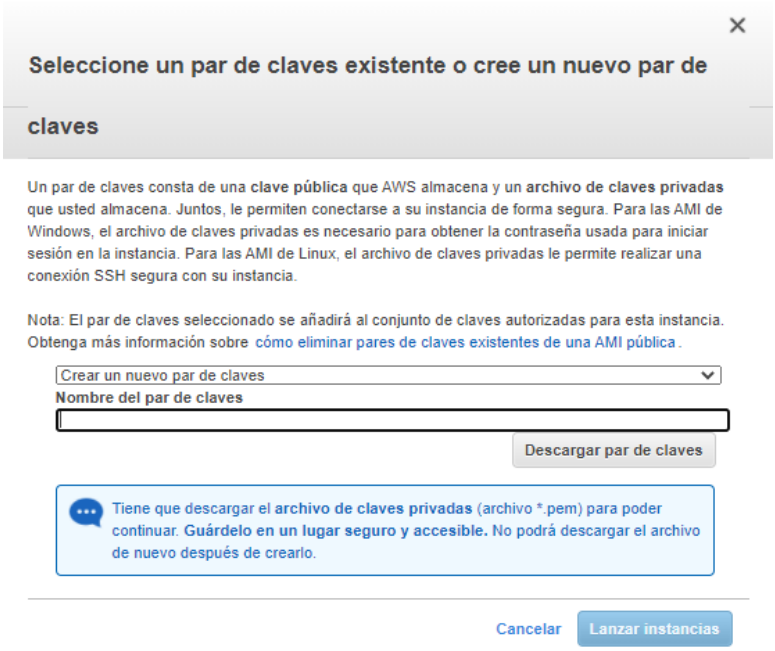


Figura 3-5:Par de claves para conectarse por SSH.

Finalmente, después de lanzada la instancia se puede comprobar el estado en el que se encuentra dicha instancia. En este caso el estado es **En ejecución** como se observa en la **Figura 3-6**, por lo que la máquina se encuentra disponible para realizar la conexión mediante SSH.

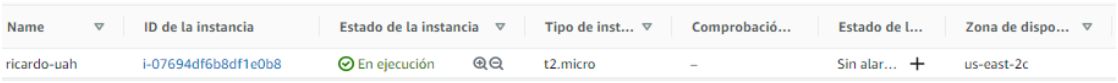


Figura 3-6: Instancia ricardo-uah en ejecución.

3.3. Conectar la instancia mediante SSH.

Como se ha indicado en el punto anterior, una vez creada la instancia se debe de acceder mediante SSH y es en este momento donde se necesita el fichero con el par de claves que se ha descargado en el paso previo. Al estar utilizando una máquina con sistema operativo Windows 10, es necesario hacer uso del software **PuTTY**, para poder realizar la conexión SSH. **PuTTY** es un emulador de terminal que admite diferentes protocolos de red como TCP (Transmission Control Protocol), Telnet, rlogin y SSH.

Para poder realizar la descarga del software, se ha accedido a la página web oficial del fabricante [21] y en ella se ha procedido a realizar la descarga de **PuTTY** para Windows. Una vez instalado el software se visualizará como en la **Figura 3-7**:

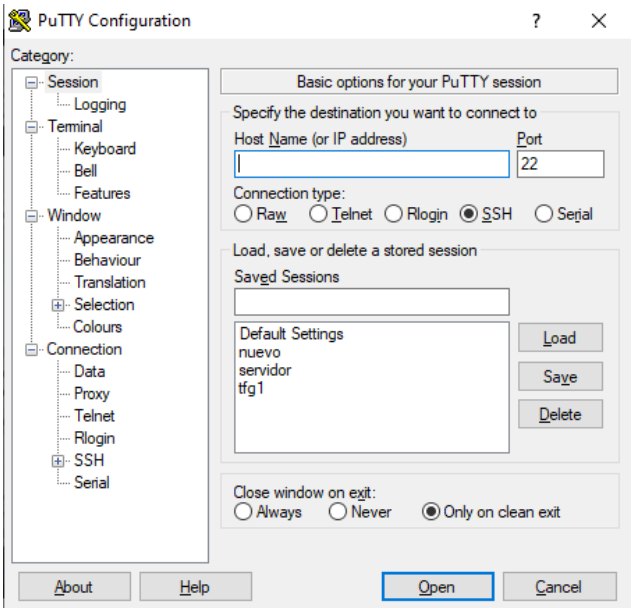


Figura 3-7: Software PuTTY.

Como el fichero del par de claves que se ha descargado de AWS cuenta con extensión **.pem** y el software **PuTTY** no acepta dicho formato, es necesario hacer uso del componente **PuTTY Key Generator**, el cual permite modificar la extensión del fichero de **.pem** a **.ppk**.

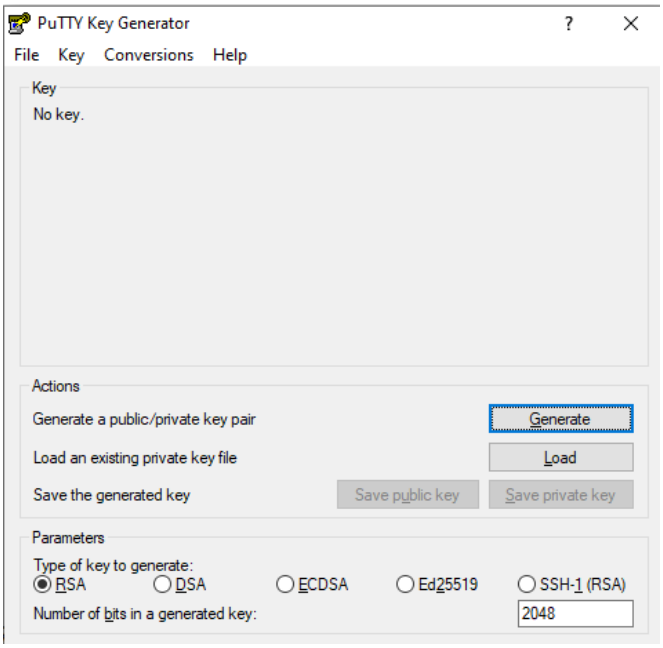


Figura 3-8: Componente PuTTY Key Generator.

En esta herramienta se procede a cargar el fichero que se desea modificar, para después seleccionar la opción *guardar la clave privada*, la cual proporciona el fichero con extensión **.ppk**. Con el archivo modificado, se puede proceder con el software de PuTTY a acceder a la instancia creada mediante SSH, para ello se selecciona el DNS (Domain Name System) público que proporciona Amazon de la máquina virtual, en este caso: **ec2-52-15-158-135.us-east-2.compute.amazonaws.com.**

En el software de **PuTTY** se debe acceder al apartado: **Conexiones -> SSH -> Auth**, que es donde se procederá a cargar el fichero de clave privada en formato **.ppk**. Una vez cargado, se debe de acceder al apartado **Sesión**, en el recuadro Host Name se indica el DNS público que se ha mencionado anteriormente y se indica como Puerto el número 22. Con esta configuración se realizará la conexión a la instancia por SSH de forma exitosa, como se muestra a continuación en la **Figura 3-9**:

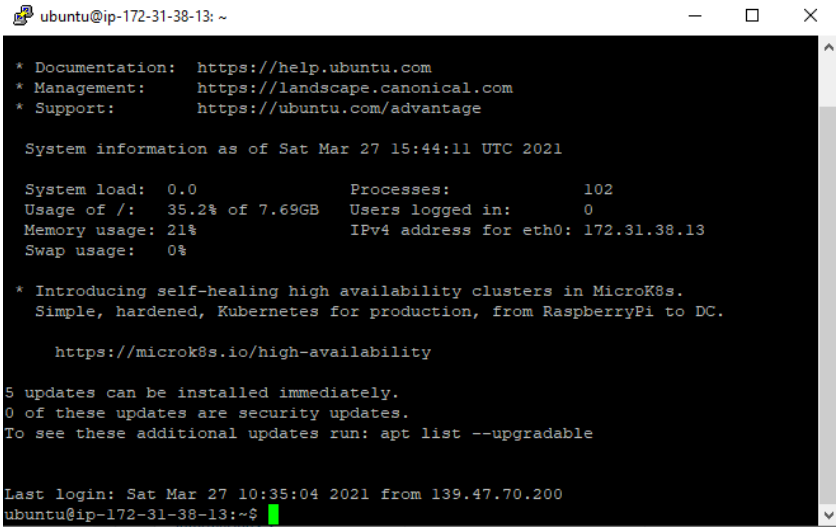


Figura 3-9: Instancia conectada por SSH.

3.4. Instalación de los paquetes básicos de Ubuntu.

Una vez dentro de la máquina virtual de Amazon, se procede a realizar la instalación de los paquetes básicos de Ubuntu. Este procedimiento se debe de realizar para garantizar que la instancia tenga todos los paquetes esenciales instalados.

Para empezar, se debe de actualizar los paquetes disponibles en la máquina virtual, esto se realiza con el comando: **sudo apt-get update && sudo apt-get upgrade**. Una vez actualizados los paquetes, se debe proceder a instalar el paquete *aptitude*, el cual sirve para mejorar la experiencia a la hora de instalar paquete, para ello se usa el siguiente comando: **sudo apt-get install aptitude.** A continuación, se realizará la instalación de los paquetes básicos de la siguiente forma: **sudo aptitude install build-essential.**

Objetivo del comando	Comandos
Actualizar los paquetes disponibles en la instancia.	sudo apt-get update && sudo apt-get upgrade
Instalar paquete aptitude.	sudo apt-get install aptitude
Instalar los paquetes básicos.	sudo aptitude install build-essential

Tabla 3-1: Resumen de comandos para instalar paquetes básicos Ubuntu.

Terminada la configuración de los paquetes esenciales, se debe comprobar que la instancia creada cuenta con Python3 instalada, la comprobación se realiza ejecutando el comando **python3 -V**, lo que devuelve la versión de Python disponible en la máquina virtual.

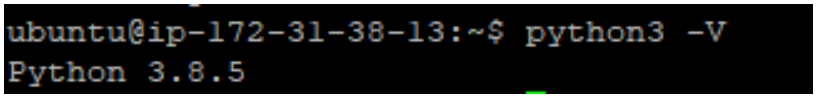


Figura 3-10: Comprobación de la versión de Python 3.

Para finalizar este punto, se debe de instalar la herramienta **pip**, la cual permite administrar paquetes de Python de forma gratuita, dicho administrador es de código abierto (open source) y multiplataforma. Al estar utilizando Python3 se puede usar la versión **pip3** que permite instalar los módulos de dicha versión.

La instalación se realiza mediante el comando: **sudo apt update && sudo apt install python3-pip**. Una vez instalado, se puede comprobar que la versión fue instalada correctamente de la siguiente forma: **pip3 --version.**

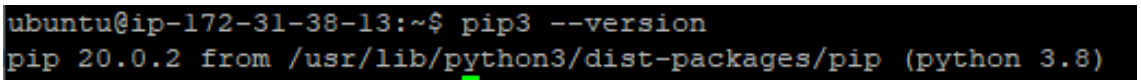


Figura 3-11: Comprobación de la instalación de pip3.

Objetivo del comando	Comandos
Comprobación de la versión de Python 3.	python3 -V
Actualizar los paquetes e instalar el paquet9e de la herramienta pip3.	sudo apt update && sudo apt install python3-pip
Comprobación de la instalación de pip3.	pip3 --version

Tabla 3-2: Resumen de comandos para la instalación de pip3.

3.5. Software Development Kit (SDK).

El SDK o traducido al español Kit de Desarrollo de Software, es un paquete de herramientas y datos que facilita a los programadores el desarrollo de programas o de una aplicación. Por lo general, los SDK son de uso gratuito, aunque el fabricante puede limitarlo con ciertas licencias.

La composición de un SDK depende del fabricante y si se encuentra diseñado para un lenguaje de programación, sistema operativo o hardware concreto. Sin embargo, un componente estándar es la API (Application Programming Interfaces o en español Interfaz de Programación de Aplicaciones), el cual se puede interpretar como un conjunto de definiciones y protocolos que se utilizan a la hora de desarrollar e integrar el software de las aplicaciones, es decir, permite la comunicación entre dos aplicaciones mediante un conjunto de reglas.

En el caso de Amazon Rekognition se encuentran diferentes SDKs, los cuales permiten simplificar el uso de los algoritmos de reconocimiento artificial en las

aplicaciones, mediante una API adaptada a la plataforma o al lenguaje de programación. Como se puede observar en la **Figura 3-12**, Rekognition cuenta con los siguientes SDK:

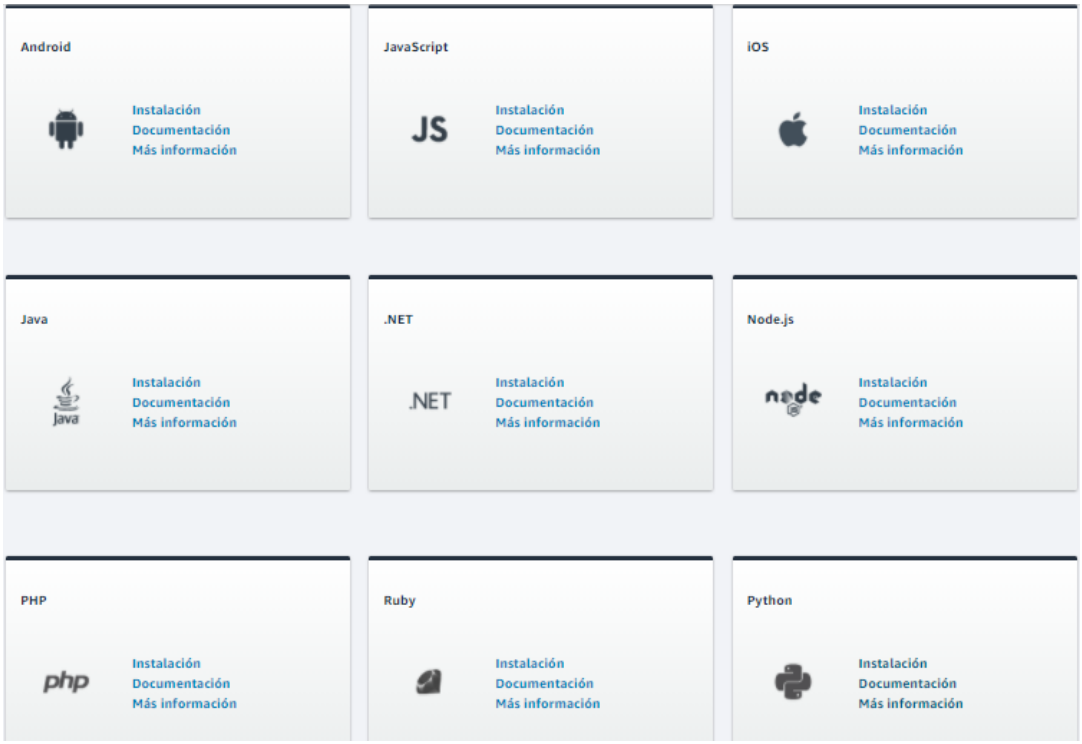


Figura 3-12: SDKs de Amazon Rekognition. *[22]***.**

Para este trabajo, se hace uso del SDK de Python, debido a que es uno de los lenguajes más comunes de aprender a lo largo de una carrera universitaria, por lo que cualquier persona se puede sentir más cómoda al momento de entender y configurar los programas que sean necesarios.

El SDK de Python para Amazon Rekogntion es **Boto3**, el cual es un paquete de Python que proporciona interfaces para AWS (Amazon Web Services). La instalación se realiza de forma muy sencilla, se puede realizar vía pip o mediante la fuente, clonando el GitHub de boto. Sin embargo, como la instancia creada cuenta con la herramienta *pip*, se procede a realizar la instalación mediante esta vía, con el comando: **pip3 install boto3.**

A continuación, se procede a realizar la instalación de AWS CLI (Command Line Interface) versión 2 para Linux, lo que permitirá hacer uso en la instancia de los comandos de AWS. El único requisito para realizar la instalación es contar con el paquete *unzip*, el cual se instala con el comando **sudo apt-get install unzip**. Para completar la instalación de AWS CLI se debe de ejecutar los siguientes comandos:

- **curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"**
- **unzip awscliv2.zip**
- **sudo ./aws/install**

```
ubuntu@ip-172-31-38-13:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

Figura 3-13: Instalación de AWS CLI.

Se puede realizar la comprobación de que se ha instalado de forma correcta ejecutando: **aws --version,** como se muestra en la **Figura 3-14.**

```
ubuntu@ip-172-31-38-13:~$ aws --version
aws-cli/2.1.32 Python/3.8.8 Linux/5.4.0-1041-aws exe/x86_64.ubuntu.20 prompt/off
```

Figura 3-14: Comprobación de la instalación AWS CLI.

Objetivo del comando	Comandos
Instalación del SDK, Boto3.	pip3 install boto3
Instalar el paquete unzip.	sudo apt-get install unzip
Descargar el paquete codificado del AWS CLI.	curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
Decodificar el fichero descargado.	unzip awscliv2.zip
Instalar el paquete del AWS CLI.	sudo ./aws/install
Comprobar la instalación del AWS CLI.	aws --version

Tabla *3-3*: Resumen instalación de SDK y AWS CLI.

3.6. Creación de un usuario en el servicio IAM de AWS.

Identity and Access Management (IAM) de AWS, permite administrar el acceso a los servicios y recursos de Amazon Web Services de forma segura. Este servicio permite crear y administrar tanto usuarios como grupos de AWS, al igual que permite, mediante permisos conceder o denegar el uso de los recursos de Amazon Web Services (gracias a la autenticación), al igual que indicar cuales recursos se pueden utilizar y de qué forma (autorización). [23]

Se puede trabajar con AWS IAM mediante las siguientes maneras:

- Consola de administración de AWS: Mediante la consola de AWS, se pueden administrar los recursos de IAM como de AWS.

- Herramientas de línea de comandos de AWS: Con la línea de comandos de AWS (AWS CLI), se pueden configurar los comandos para realizar tareas de IAM y Amazon Web Services.
- SDK de AWS: Los SDK se encargan de crear acceso programático a IAM y AWS, realizando tareas como la firma encriptada de solicitudes, administrar errores y reintentar solicitudes de forma automática.
- API IAM HTTPS: Se puede acceder a IAM y AWS mediante programación de la API IAM HTTPS (Hypertext Transfer Protocol Secure), que permite generar solicitudes HTTP (Hypertext Transfer Protocol) directamente al servicio. El uso de la API HTTPS, debe de incluir código para realizar la firma digital de las solicitudes con las credenciales del usuario.

Para hacer uso de la SDK se debe de crear un usuario IAM que permita, mediante un ID de acceso y una clave secreta hacer uso del Software Development Kits, y por ende de la línea de comandos y de la API de AWS.

Para crear el usuario IAM, se debe de dirigir en la consola de AWS, al servicio **IAM** (Identity and Access Management), en la sección *Administración del acceso*, seleccionar la opción *Usuario* y se debe de seleccionar el botón *Añadir usuario(s).* Se procederá a escribir el nombre del usuario(s) a crear y para el tipo de acceso de AWS, se selecciona la opción *Acceso mediante programación*, la cual permite hacer uso del SDK, la CLI, la API y otras herramientas de desarrollo.

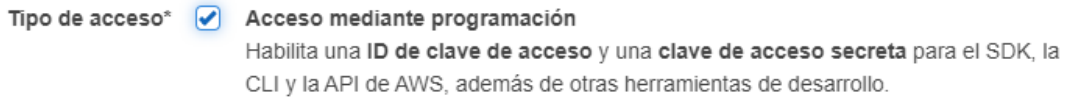


Figura 3-15: Tipo de acceso del usuario IAM.

Al momento de crear el usuario, se debe de asignar políticas al usuario, al tratarse de Rekognition se encuentran 4 posibles permisos:

- AmazonRekognitionServiceRole: Esta política especifica los permisos de Rekognition y S3 requeridos por las etiquetas personalizadas de Amazon Rekognition.
- AmazonRekogntionReadOnlyAccess: Acceso a todas las API de Amazon Rekognition.
- AmazonRekognitionFullAccess: Acceso a todas las API de reconocimiento de lectura.
- AmazonRekognitionCustomLabelsFullAccess: Permite que Rekognition llame a los servicios de AWS en tu nombre.

Figura 3-16: Permisos de Rekognition a usuario IAM.

Sin embargo, para la realización de este trabajo, al estar realizando las funciones con un usuario root, se ha seleccionado el permiso llamado **AdministratorAccess**, el cual proporciona acceso completo a los servicios y recursos de AWS.

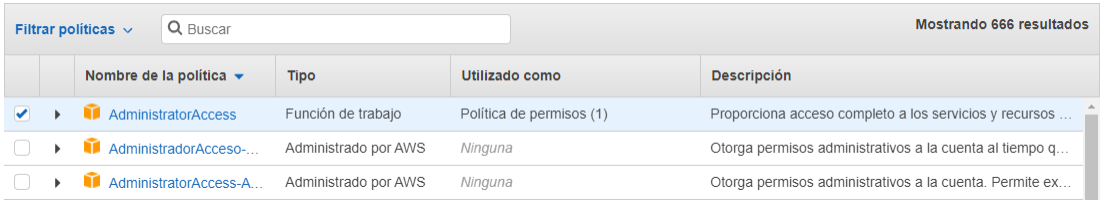


Figura 3-17: Permiso AdministratorAccess.

Con los permisos otorgados, se procede a descargar las credenciales del usuario que se ha creado, de igual forma dicha clave se puede observar en la consola de AWS siempre que el usuario desee.

Figura 3-18: Credenciales del usuario IAM.

Terminada la configuración del usuario IAM, con sus respectivas credenciales, se procede a ingresar dichas claves en la máquina virtual, haciendo uso de la línea de comandos de AWS, conocida como AWS CLI. Para ello es necesario ejecutar el comando: **aws configure --profile ricardo,** creando de esta forma el perfil del usuario, por lo que se deben de asignar tanto el ID como la clave de acceso del usuario IAM, se indica la región de la instancia e indicando como formato de salida JSON, como se detalla en la **Figura 3-19**.

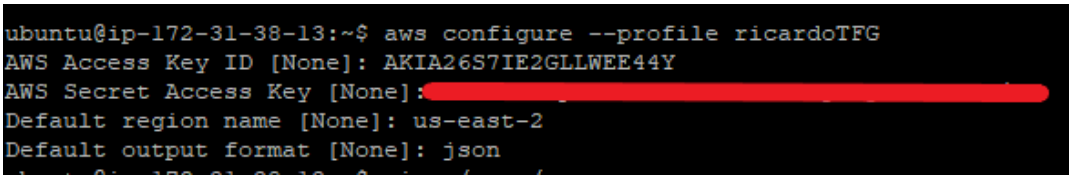


Figura 3-19: Configuración de las credenciales.

Para comprobar que el perfil se ha configurado correctamente, se puede acceder al fichero **~/.aws/,** donde se observa el perfil con la región de la instancia y el formato de salida, como se observa en la **Figura 3-20**.

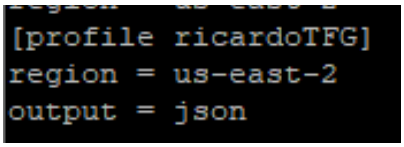


Figura 3-20: Configuración fichero ~/.aws/.

4. Reconocimiento de objetos con AWS Rekognition.

En este capítulo se procede con el desarrollo de los programas en lenguaje Python necesarios para hacer uso de los algoritmos de reconocimiento de objetos de Amazon Rekognition.

Con los programas que se han creado, se podrá reconocer todos los objetos de la imagen con el detector de Amazon. De igual forma, mediante el uso de las etiquetas personalizadas de Rekognition, se ha creado el modelo, el cual se debe de entrenar para que pueda reconocer los objetos mediante las etiquetas creadas.

4.1. Reconocimiento de objetos en una imagen.

En este apartado, se realiza el estudio de la herramienta de Rekognition, haciendo uso de la función de reconocimiento que devuelve en formato JSON todos los objetos que se identifiquen en la imagen.

4.1.1. Desarrollo de los programas de Python.

Se han desarrollado los programas de Python necesarios para hacer uso de Amazon Rekognition. Los programas que se desarrollaron fueron 2, el primer programa permite transformar una imagen recibida mediante URL en bytes, lo que permitirá realizar el análisis de objetos; en el segundo programa, mediante el uso del módulo SDK se podrá analizar la imagen y se obtienen como respuestas los objetos que Amazon Rekognition haya identificado.

Programa: image.py.

El primer programa se ha llamado **image.py** (el contenido del programa se encuentra en el **Anexo I,** apartado **A.1.**), ha sido necesario este programa de Python para contar con acceso a diferentes imágenes, mediante el uso de una URL. Para la creación del programa se ha utilizado el editor de texto **vim**, por lo que se ha usado el siguiente comando: **vim image.py**.

La función del programa recibe como parámetro la URL de la imagen y devolverá la imagen en bytes, que es el formato en el que se necesita enviar la imagen para que el SDK de Rekognition pueda analizar la imagen y detectar los objetos que identifiquen en ella. De la misma forma, se puede modificar este programa para obtener en bytes los datos de una imagen que se encuentre en el servidor, con la finalidad de evitar que el

reconocimiento de imágenes dependa de que la URL de la imagen se encuentre disponible.

Programa: demo.py.

El segundo programa se ha llamado **demo.py** (el contenido del programa se encuentra en el **Anexo I**, apartado **A.2**.), se ha utilizado el editor de texto **vim** para la creación, mediante el comando **vim demo.py**.

Este programa permite, mediante el uso del SDK de Python, conectarse con la API de Rekognition, para enviarle los datos de la imagen. Al usar imágenes de Internet es necesario hacer uso del programa anterior (**image.py**), la API retornará como respuesta, los datos que se hayan sido identificado en la imagen.

La función del módulo boto3 que se ha utilizado es *detect_labels*, la cual permite detectar instancias de entidades del mundo real dentro de una imagen en formato JPEG o PNG. Es por ello que se envía como parámetro de la función la imagen en bytes, dicha función también permite enviar como parámetro un máximo de etiquetas a analizar o garantizar que las etiquetas recibidas cuenten con un mínimo de confianza, de esta forma se registrarán únicamente las etiquetas con mayor fiabilidad.

```
response = client.detect_labels(
    Image={
        'Bytes': b'bytes',
        'S3Object': {
            'Bucket': 'string',
            'Name': 'string',
            'Version': 'string'
        }
    },
    MaxLabels=123,
    MinConfidence=...
)
```

Figura 4-1: Sintaxis de detect_labels. *[24]***.**

Como se ha comentado anteriormente, la función acepta 3 parámetros, siendo el primero el obligatorio porque es la imagen en bytes, el segundo parámetro es el máximo de etiquetas y el tercero es el mínimo de confianza aceptable, tal cómo se puede observar en la **Figura 4-1**. La respuesta de la función se obtiene en formato JSON, como se visualiza a continuación:

Figura 4-2:Respuesta de la función detect_labels en JSON.

En la **Figura 4-2**, se observan 2 etiquetas detectadas (Sedan y Car), de igual forma el nombre, la lista con las etiquetas principales, incluyendo los ancestros, el nivel de confianza del sistema con respecto al objeto identificado y las coordenadas del objeto para poder crear un cuadro que delimite la ubicación del objeto detectado.

El programa **demo.py**, ha sido diseñado para que, mediante un *for*, se muestre por pantalla todas las etiquetas recibidas con el nivel de confianza que le corresponde, notificando de esta forma los objetos detectados por la API de Rekognition.

4.1.2. Reconocimiento de objetos con Rekognition.

Con los dos programas que se han creado, se puede empezar a reconocer objetos gracias al servicio Amazon Rekognition. Para ello se han utilizado 4 imágenes recogidas de Internet, facilitando de esta forma la comprobación de los datos que se reciben en el programa **demo.py** con el detector de objetos que se encuentra disponible en la consola de AWS.

La primera imagen que se utiliza es la de un coche eléctrico y una chica, también se encuentran otros coches aparcados en la calle, tal como muestra la **Figura 4-3**.

Figura 4-3: Imagen de coche y chica. *[25]*.

Para empezar, se ha utilizado el detector de la página de Amazon Rekognition para identificar los objetos que esta detecta. En la **Figura 4-4** se observa que reconoce, mediante diversas etiquetas, varios coches y a una persona.

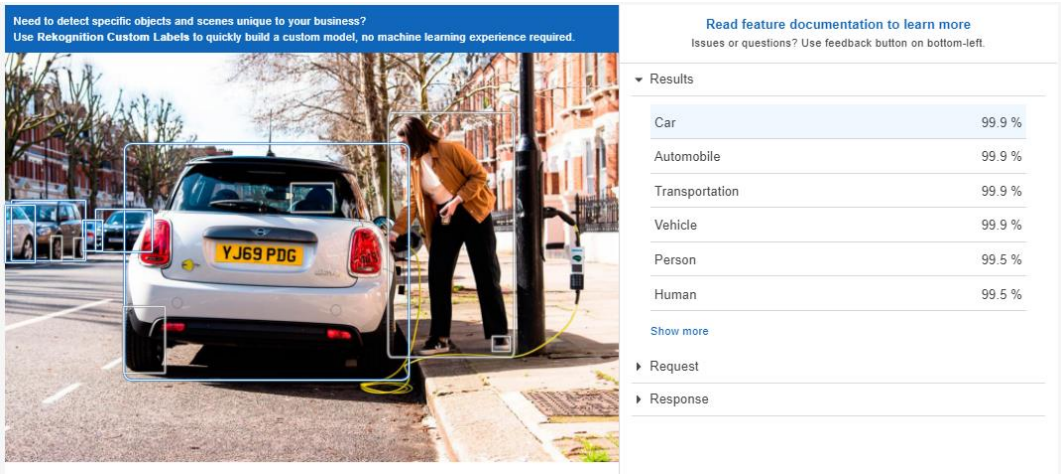


Figura 4-4: Resultados de la imagen en la página web de Amazon Rekognition.

Con las detecciones encontradas por el detector de Amazon, se procederá a ejecutar el programa **demo.py**, el cual deberá de mostrar todas las etiquetas que aparecieron en el detector e incluso más.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 demo.py
Esto es lo que observamos de la imagen:
Car con un porcentaje del:  99.94957733154297 %
Automobile con un porcentaje del:  99.94957733154297 %
Transportation con un porcentaje del:  99.94957733154297 %
Vehicle con un porcentaje del:  99.94957733154297 %
Person con un porcentaje del:  99.52592468261719 %
Human con un porcentaje del:  99.52592468261719 %
Machine con un porcentaje del:  86.65974426269531 %
Wheel con un porcentaje del:  81.76557159423828 %
Tire con un porcentaje del:  79.88532257080078 %
Shoe con un porcentaje del:  70.86258697509766 %
Clothing con un porcentaje del:  70.86258697509766 %
Footwear con un porcentaje del:  70.86258697509766 %
Apparel con un porcentaje del:  70.86258697509766 %
Alloy Wheel con un porcentaje del:  65.03142547607422 %
Spoke con un porcentaje del:  65.03142547607422 %
Car Wheel con un porcentaje del:  64.2005615234375 %
License Plate con un porcentaje del:  62.88629913330078 %
Sports Car con un porcentaje del:  57.730899810791016 %
```

Figura 4-5: Resultados obtenidos por el programa demo.py.

Para esta segunda prueba se utiliza una imagen de dos semáforos (**Figura 4-6**). Como se muestra en la **Figura 4-7**, el detector de la consola de AWS detecta tanto el semáforo como la luz en rojo del semáforo.

Figura 4-6: Imagen semáforos. *[26]***.**

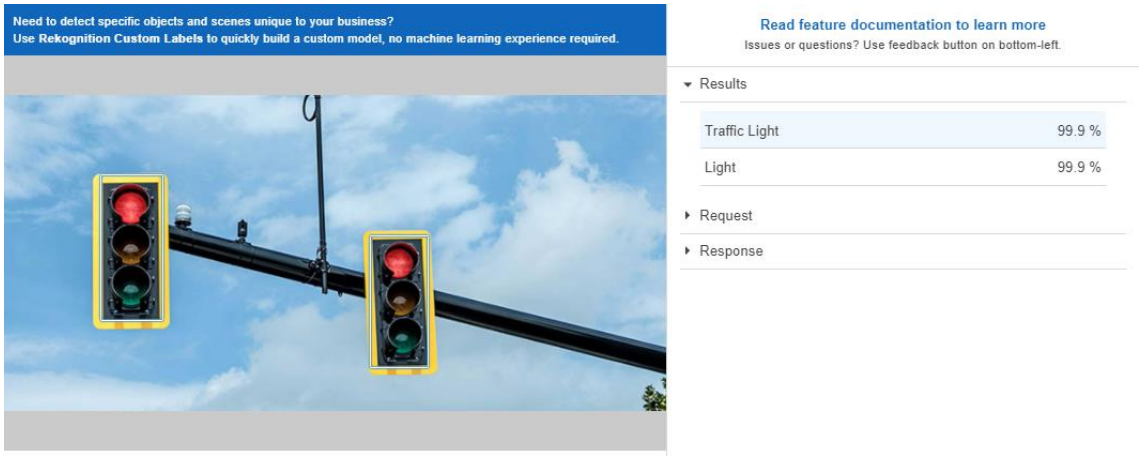


Figura 4-7: Resultados de la imagen en la página web de Amazon Rekognition.

Al ejecutar el programa **demo.py,** con la imagen de los dos semáforos se observa que se obtuvo las mismas etiquetas que el detector de la consola AWS y con el mismo porcentaje de precisión:

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 demo.py
Esto es lo que observamos de la imagen:
Traffic Light con un porcentaje del:  99.9962158203125 %
Light con un porcentaje del:  99.9962158203125 %
```

Figura 4-8: Resultados obtenidos por el programa demo.py.

Como tercera imagen se ha optado por usar una imagen de un hidratante o boca de incendio, como se observa en la **Figura 4-9**.

Figura 4-9: Imagen de boca de incendio. *[27]*.

Tanto el detector web de Amazon como el programa **demo.py**, detectan la boca de incendio sin problema y con el mismo porcentaje de precisión. Tal como se muestra a continuación en la **Figuras 4-10** y **4-11**:

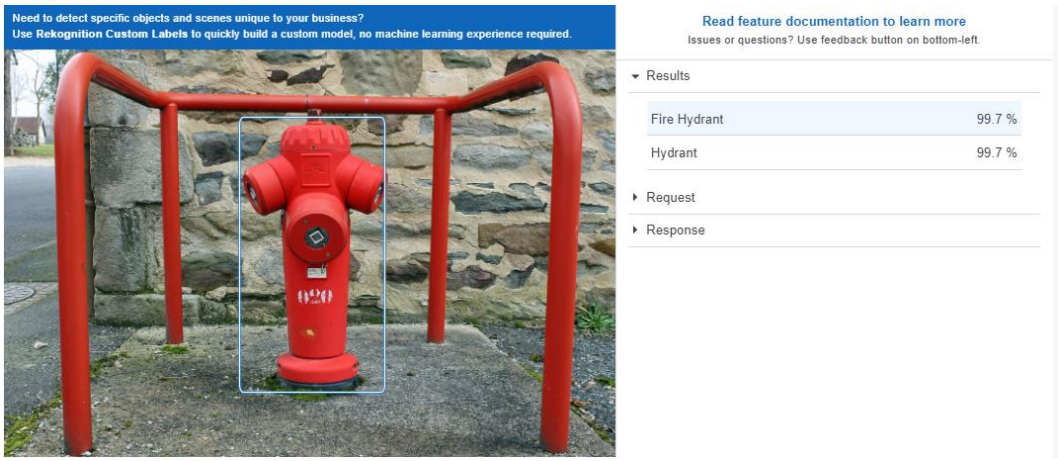


Figura 4-10: Resultados de la imagen en la página web de Amazon Rekognition.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 demo.py
Esto es lo que observamos de la imagen:
Fire Hydrant con un porcentaje del:  99.7092514038086 %
Hydrant con un porcentaje del:  99.7092514038086 %
```

Figura 4-11: Resultados obtenidos por el programa demo.py.

Para finalizar, se ha utilizado una imagen de un dormitorio, donde hay muchos objetos para identificar por Rekognition, como se observa en la **Figura 4-12**.

Figura 4-12: Imagen de dormitorio. *[28]*.

En el detector web de Amazon y en el programa **demo.py**, se detectan muchos objetos, aunque como se puede observar en las **Figuras 4-13** y **4-14**, faltan algunos elementos de la imagen por identificar.

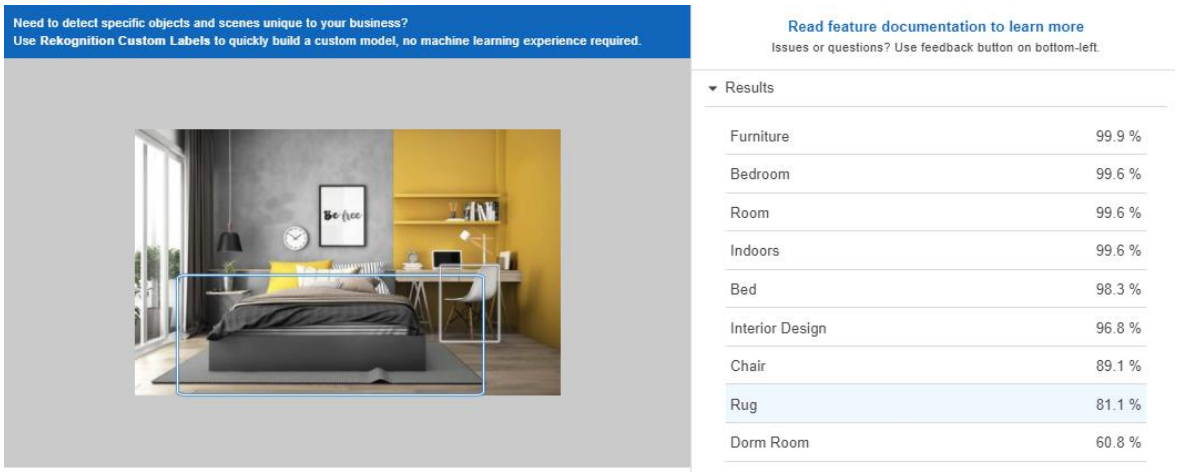


Figura 4-13: Resultados de la imagen en la página web de Amazon Rekognition.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 demo.py
Esto es lo que observamos de la imagen:
Furniture con un porcentaje del:  99.97405242919922 %
Bedroom con un porcentaje del:  99.6692886352539 %
Room con un porcentaje del:  99.6692886352539 %
Indoors con un porcentaje del:  99.6692886352539 %
Bed con un porcentaje del:  98.3858642578125 %
Interior Design con un porcentaje del:  96.80469512939453 %
Chair con un porcentaje del:  89.19874572753906 %
Rug con un porcentaje del:  81.17764282226562 %
Dorm Room con un porcentaje del:  60.888999938964844 %
Pillow con un porcentaje del:  56.945762634277344 %
Cushion con un porcentaje del:  56.945762634277344 %
```

Figura 4-14: Resultados obtenidos por el programa demo.py.

En base a los resultados obtenidos para las 4 imágenes analizadas, la precisión de Amazon Rekognition es bastante alta cuando en la imagen se encuentran pocos objetos. Sin embargo, cuando se realizan detecciones en fotos con múltiples objetos se observa que, a pesar de identificar correctamente a la mayoría , el nivel de confianza del algoritmo de reconocimiento entrenado no es tan alto. Al igual que pasa en el último análisis (**Figura 4-12**), hay objetos, como el cuadro y los libros que no se detectan.

Para finalizar, este servicio de visión artificial no permite detectar un objeto en específico en la imagen. Cuando se desee realizar esta acción, se deben de usar las etiquetas personalizadas de Amazon.

4.2. Reconocimiento de objetos con etiquetas personalizadas.

En este apartado se realiza el análisis de imágenes mediante el uso de etiquetas personalizadas, las cuales permiten identificar objetos y escenas en fotos muy específicas, lo que resulta muy útil porque permite entrenar el algoritmo de reconocimiento.

Las etiquetas personalizadas permiten identificar productos en un lugar, clasificar piezas de máquinas en una línea de ensamblaje, distinguir plantas sanas e infectadas o detectar personajes animados en vídeos.

Las etiquetas se basan en las capacidades existentes de Rekognition, debido a que ya están capacitadas por decenas de millones de imágenes en muchas categorías. Gracias a ello, es necesario cargar únicamente un pequeño conjunto de imágenes de entrenamiento, permitiendo producir un modelo de análisis de imágenes personalizadas en pocas horas. Si el dataset de imágenes ya se encuentra en Rekognition, el entrenamiento del modelo se puede realizar en poco tiempo.

Se puede comenzar a utilizar las etiquetas personalizadas de Amazon Rekognition gracias a una prueba gratuita durante 3 meses, la cual incluye 10 horas de entrenamiento de modelo gratuitas al mes y 4 horas de interferencias gratuitas al mes.

4.2.1. Preparar los datos de entrenamiento.

Para este punto, se debe de recopilar las imágenes que contienen los objetos, escenas y conceptos específicos que se desean identificar. Para ello, se debe de crear un proyecto que permita administrar los archivos del modelo, se entrena y evalúa el modelo para poder dejarlo listo para su uso.

Amazon recomienda usar como mínimo 10 imágenes para poder generar una etiqueta. Sin embargo, mientras más imágenes se tengan, el modelo tendrá más confianza al momento de realizar la detección. Las imágenes a cargar deben de estar en formato JPEG o PNG.

Para la recopilación de las imágenes es necesario hacer uso de un bucket en Amazon S3, que es un contenedor de objetos (archivos) en S3. En el caso de este trabajo, se le ha asignado el nombre de **demo-libros-llaves** al bucket, donde se han almacenado 34 imágenes de libros y 17 imágenes de llaves, lo que permitirá crear dos etiquetas, una para las llaves y otra para los libros.

Con las imágenes cargadas en el bucket, se debe acceder mediante la consola de AWS. al servicio Amazon Rekognition y seleccionar la opción *Use Custom Labels* (que

en español significa, use etiquetas personalizadas). Al seleccionar dicha opción, Rekognition solicitará crear un nuevo bucket en Amazon S3, para guardar los ficheros del proyecto.

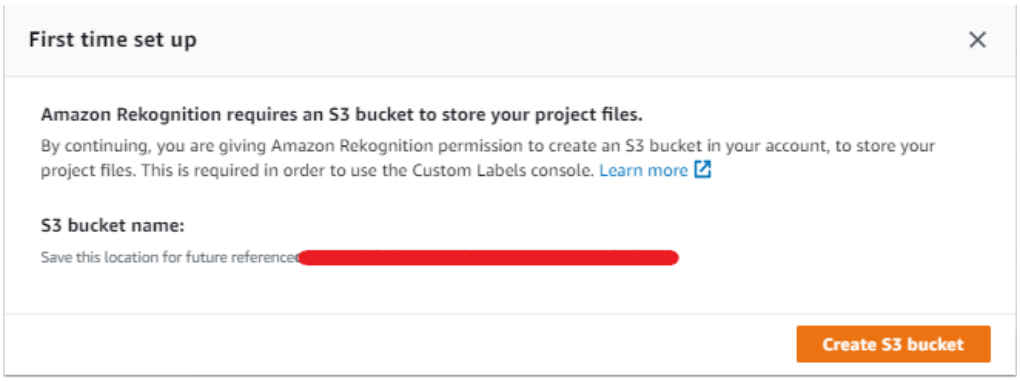


Figura 4-15: Solicitud de creación de bucket para el proyecto.

Una vez creado el bucket, se procede a crear el proyecto, para ello se le ha asignado el nombre **demo-project**. A continuación, se procederá a cargar el conjunto de datos que se utilizará para la creación de la(s) etiqueta(s) personalizada(s), para ello hay 4 opciones que permiten proporcionar los datos de entrenamiento:

- Importar imágenes etiquetadas por SageMarker Ground Truth.
- Importar imágenes desde el depósito S3.
- Copiar un conjunto de datos de etiquetas personalizadas de Rekognition existentes.
- Subir imágenes desde el ordenador.

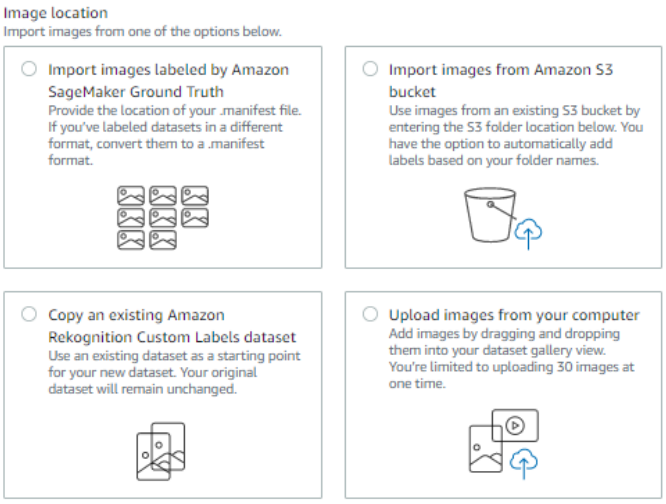


Figura 4-16: Localización de las imágenes.

En este caso, las imágenes de entrenamiento se encontraban guardadas con anterioridad en el bucket **demo-libros-llaves**, por lo que se selecciona la opción de importar imágenes desde el cubo de Amazon S3, se debe de escribir la dirección de dicho bucket. Para finalizar, se debe de copiar las políticas que genera Amazon en el bucket de las imágenes para poder continuar, de esta forma las imágenes estarían importadas.

4.2.2. Etiquetar los datos de entrenamiento.

Con los datos de entrenamiento configurados, se debe proceder a etiquetar las imágenes para de esta forma poder realizar el entrenamiento del modelo.

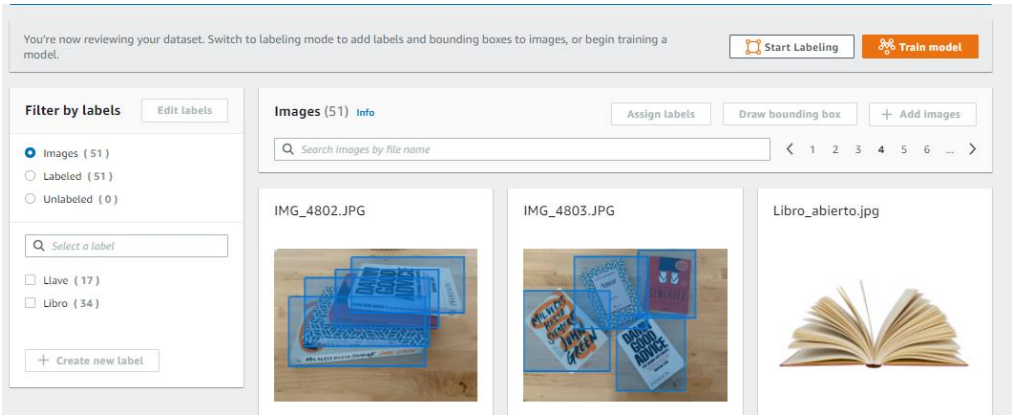


Figura 4-17: Etiquetado de las imágenes.

Para ello, se debe de crear una etiqueta por cada elemento que se desea identificar, en el caso de este apartado, se crearon 2 etiquetas una llamada Libro y otra Llave. De igual forma, Rekognition permite importar una lista de etiquetas existentes dentro del dataset.

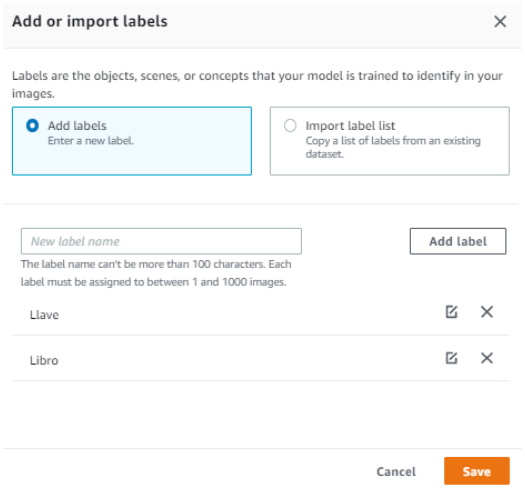


Figura 4-18: Creación de las etiquetas.

Para realizar el etiquetado de las imágenes existen dos opciones, como se observa en la **Figura 4-19**. La primera es útil cuando el objeto se encuentra solo porque es fácil de seleccionar por lo que se puede seleccionar la imagen y relacionarla con la etiqueta creada para que de esta forma se pueda identificar el objeto. La segunda opción es útil cuando el objeto se encuentra con más objetos, por lo que se puede crear un marco pequeño que delimite al objeto y que de esta forma se pueda identificar.

Figura 4-19: Tipos de etiquetado.

Las imágenes se deben ir seleccionando de una en una y asignándoles las etiquetas a cada una de ellas, para finalizar se deben de guardar los cambios realizados. Se puede comprobar que el etiquetado se realizó correctamente cuando se indica que todas las imágenes se encuentran etiquetadas.

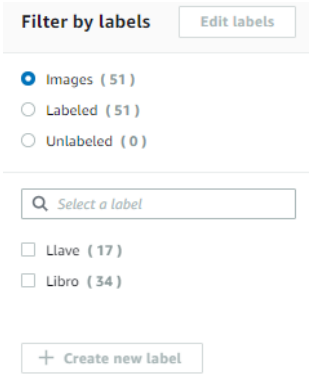


Figura 4-20: Imágenes etiquetadas.

4.2.3. Entrenamiento del modelo.

Una vez asignadas todas las imágenes a su respectiva etiqueta llega el momento de entrenar al modelo. Para ello se debe de seleccionar el botón **Train Model**, como se indica en la **Figura 4-21**:

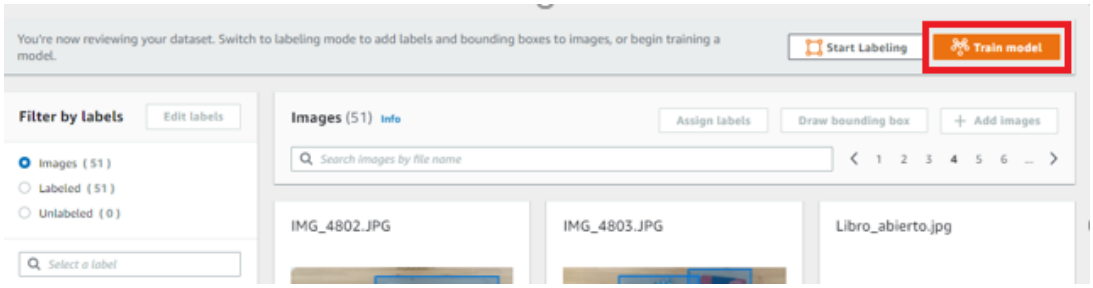


Figura 4-21: Entrenar el modelo.

Para continuar, el sistema pide seleccionar el dataset que se ha creado anteriormente y luego solicita un tipo de entrenamiento. La opción a utilizar cuando se está creando un modelo es la de *Split training dataset*, donde el 20% de las imágenes guardadas se utilizan para realizar pruebas al modelo, mientras que el 80% restante de las imágenes son utilizadas para entrenar el modelo.

Al seleccionar esta opción, el proyecto empieza a realizar el entrenamiento como se muestra en la **Figura 4-22**:

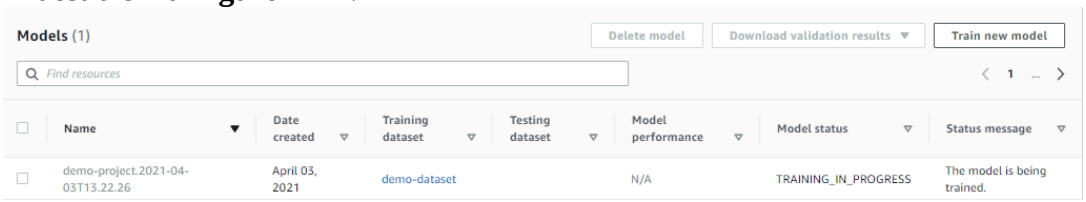


Figura 4-22: Entrenamiento en progreso.

El proceso de entrenamiento del modelo tardará dependiendo de la cantidad de imágenes que se le proporcionen, en este caso la duración fue de 1 hora. Es importante recordar que el servicio de entrenamiento tiene un coste de 1$ por hora.

Una vez finalizado el entrenamiento, el estado del modelo se actualizará indicando que se encuentra listo para su uso y se puede observar el rendimiento general del mismo, es decir, el promedio para todo el conjunto de etiquetas es de 0.762. Esta medida tiene en cuenta tanto el nivel de precisión como la recuperación de todas las etiquetas, por lo que el valor oscila entre 0 y 1.

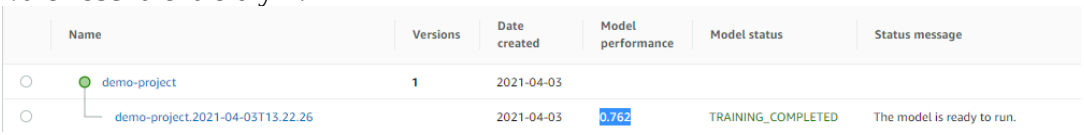
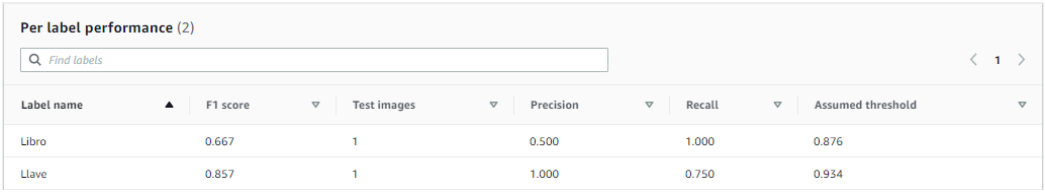


Figura 4-23: Entrenamiento completado.

De igual forma, Amazon permite acceder al detalle del modelo donde se especifican los resultados para cada una de las etiquetas creadas. Para este proyecto, estos fueron los valores obtenidos:


Per label performance (2)

Find labels

1

Label name	F1 score	Test images	Precision	Recall	Assumed threshold
Libro	0.667	1	0.500	1.000	0.876
Llave	0.857	1	1.000	0.750	0.934

Figura 4-24: Puntuación de las etiquetas del modelo.

La puntuación F1 es la media armónica de las puntuaciones de precisión y recuperación en el umbral asumido. La precisión por su parte es la división de las predicciones correctas (conocidas como verdaderos positivos) sobre todas las predicciones del modelo (verdaderos y falsos positivos), en el umbral asumido por cada etiqueta. La recuperación se puede definir como la medida de la frecuencia con la que el modelo puede predecir correctamente una etiqueta personalizada cuando está realmente presente en las imágenes del conjunto de prueba.

En este caso, se puede identificar que la etiqueta Libro tiene una precisión baja pero una alta recuperación , lo que quiere decir que se predecirá correctamente más libros reales, pero hay más predicciones de libros que estarán equivocadas. Mientras que en la etiqueta Llave pasa justo lo contrario, es decir, tiene una alta precisión, pero una recuperación moderada, por lo que habría más predicciones correctas de llaves y no se perderían tantas predicciones de llaves reales, que es justamente a lo que un buen modelo debe de aspirar.

Con dichas métricas se puede conocer y trazar los pasos necesarios que permitan mejorar el rendimiento de un modelo, en caso de que sea necesario. El rendimiento de los modelos de aprendizaje depende en gran medida de la complejidad y de la variabilidad de las etiquetas personalizadas, por lo tanto, para mejorar un modelo Amazon recomienda aumentar las cantidades de datos de entrenamiento y que dichas imágenes sean de mejor calidad.

4.2.4. Reconocimiento de etiquetas personalizadas.

Para hacer uso de los modelos de aprendizaje es necesario pagar 4$ por hora para poder realizar detecciones personalizadas, lo que hace que el uso del servicio 24/7 se aplique relativamente en proyectos donde sea necesario, mientras que para aplicaciones más sencillas obliga a controlar el nivel de uso que se le está dando al modelo.

Como se ha creado con anterioridad el modelo de aprendizaje, se procede a realizar un programa que permita utilizar las Custom Labels que se han creado. Para ello es necesario hacer uso de los códigos de Python que provee Amazon, que permiten iniciar o detener el modelo. El programa **start.py**, permite la inicialización mientras que **stop.py** realiza la detección del modelo, el contenido de estos dos programas se encuentra en el **Anexo I**, en los apartados **A.3** y **A.4**, respectivamente.

De igual forma se ha creado un programa llamado **custom.py** que permite hacer uso de las etiquetas personalizadas, haciendo uso de la función *detect_custom_labels* proporcionada por el módulo *boto3* de Python.

```
response = client.detect_custom_labels(
    ProjectVersionArn='string',
    Image={
        'Bytes': b'bytes',
        'S3Object': {
            'Bucket': 'string',
            'Name': 'string',
            'Version': 'string'
        }
    },
    MaxResults=123,
    MinConfidence=...
)
```

Figura 4-25: Sintaxis de detect_custom_labels. *[24]*.

Como se observa en la **Figura 4-25**, la función necesita como parámetros obligatorios el ARN (Amazon Resources Name) del modelo que se desea utilizar y la imagen en bytes de un objeto en Amazon S3. De igual forma se puede enviar un máximo de etiquetas a visualizar y un mínimo de confianza para la detección.

Por lo tanto, para iniciar el procedimiento de detección de objetos es necesario ejecutar el primer programa llamado **start.py**, el cual realiza la conexión para iniciar el modelo. Dicho programa tarda unos minutos en ejecutarse, pero informa cuando el procedimiento se ha realizado correctamente.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 start.py
Starting model: arn:aws:rekognition:us-east-2:752892782220:project/demo-project/
version/demo-project.2021-04-03T13.22.26/1617448945639

Status: RUNNING
Message: The model is running.
Done...
```

Figura 4-26: Inicio del modelo con start.py.

Se puede comprobar que el modelo se ha iniciado correctamente en la página web de Amazon Rekognition como muestra la **Figura 4-27**.

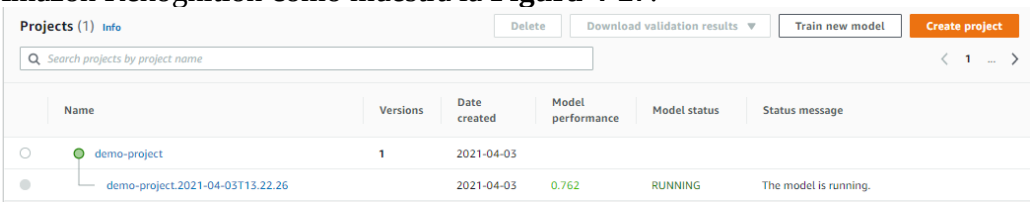


Figura 4-27: Modelo ejecutándose.

Para comprobar la correcta detección de los objetos, se han usado 2 imágenes disponibles en el bucket. En la primera se visualizan 3 llaves (**Figura 4-28**), mientras que en la segunda imagen se mezcla un libro con una llave (**Figura 4-30**).

Figura 4-28: Imagen llaves.

Al ejecutar el programa **custom.py**, se detectan las 3 llaves que se encuentran en la imagen con un alto porcentaje de confiabilidad.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 custom.py
Esto es lo que observamos de la imagen:
Llave con un porcentaje del:  97.1189956665039 %
Llave con un porcentaje del:  97.09100341796875 %
Llave con un porcentaje del:  96.39899444580078 %
```

Figura 4-29: Resultado obtenido por custom.py.

En esta segunda foto, al utilizarse las dos etiquetas se observa que después de ejecutar el programa **custom.py**, el SDK de Amazon Rekognition las detecta sin ningún problema, como se muestra en la **Figura 4-31**.

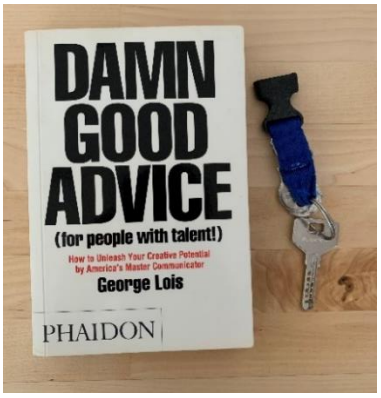


Figura 4-30: Imagen libro y llave.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 custom.py
Esto es lo que observamos de la imagen:
Libro con un porcentaje del:  99.02999877929688 %
Llave con un porcentaje del:  96.13199615478516 %
```

Figura 4-31: Resultados obtenidos por custom.py.

Para poder realizar una detección de un objeto en particular, es necesario crear un proyecto con una única etiqueta personalizada, en este caso la etiqueta Llave será la utilizada. Siguiendo los pasos anteriormente comentados, se ha creado un nuevo proyecto llamado **demo-single-object**.

Después de crear y entrenar el modelo, se han descargado los programas **ostart.py**, **ostop.py** y se ha creado el programa **ocustom.py** (los contenidos del programa se encuentran en el **Anexo I**, apartados **A.6, A.7** y **A.8**), dichos programas permiten iniciar y detener el modelo, al igual que realizar la detección del objeto etiquetado.

Los programas **ostart.py** y **ostop.py** son generados por Amazon y se utilizan para iniciar y detener, respectivamente el modelo. Mientras que el programa **ocustom.py**, se modifica únicamente el ARN del nuevo modelo, porque el resto del programa es idéntico a **custom.py**.

Para realizar la prueba de reconocimiento de objetos particulares, se ha utilizado la **Figura 4-30,** porque en ella aparecen las 2 etiquetas creadas anteriormente. Con dicha imagen se puede observar que el programa **ocustom.py** detecta únicamente la etiqueta Llave, debido a que es la única etiqueta que se ha registrado en el modelo.

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 ocustom.py
Esto es lo que observamos de la imagen:
Llave con un porcentaje del:  94.14900207519531 %
```

Figura 4-32: Resultado obtenido por ocustom.py.

Finalmente, después de obtener la detección del objeto en las pruebas realizadas se debe de ejecutar tanto el programa **ostop.py** y **stop.py** porque permiten detener los dos modelos que se han utilizado, tal y como muestra la **Figura 4-33**:

```
ubuntu@ip-172-31-38-13:~/TFG$ python3 stop.py
Stopping model:arn:aws:rekognition:us-east-2:752892782220:project/demo-project/v
ersion/demo-project.2021-04-03T13.22.26/1617448945639
Status: STOPPING
Done...
```

Figura 4-33: Detención del modelo con stop.py.

De igual forma, se puede comprobar que el modelo se ha detenido correctamente desde la página web de Amazon Rekognition, como se visualiza en la **Figura 4-34**.

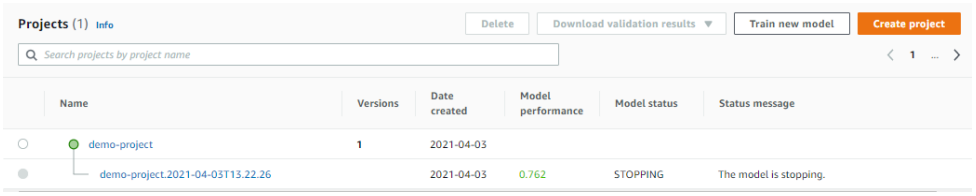


Figura 4-34: Modelo detenido.

5. Reconocimiento de objetos desde un nodo IoT/Edge.

En el capítulo 4 se realizó la implementación de los algoritmos de reconocimiento en un ordenador con conexión a Internet. Para este capítulo se pretende llevar a cabo la configuración y ejecución en un sistema de bajo consumo, con baja potencia y conectado a la red como nodo IoT, en este caso se usará una Raspberry Pi.

5.1. Instrumentos a usar.

A continuación, se describen los instrumentos necesarios para poder usar una Raspberry Pi:

- Placa Raspberry Pi Modelo B Revision 2.0.
- Fuente de alimentación para la Raspberry Pi.
- Tarjeta SD (Secure Digital) con una capacidad de 32 GB.
- Teclado y ratón USB (Universal Serial Bus).
- Monitor externo y cable HDMI (High-Definition Multimedia Interface)
- Cable Ethernet.

5.2. Configuración de la tarjeta SD.

Para poder hacer uso de la placa Raspberry Pi es necesario instalar el SO (Sistema Operativo) que se desea utilizar para la placa. Como primer paso se debe insertar la tarjeta SD en el ordenador, luego se procede a descargar el programa Raspberry Pi Imager, el cual permitirá generar la imagen que se utilizará en la Raspberry [29].

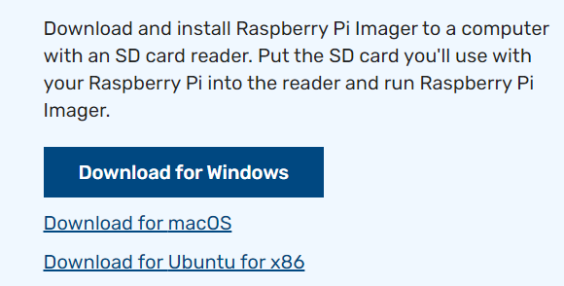


Figura 5-1: Descargar el programa según el SO del ordenador. *[29]*.

Una vez descargado el software Raspberry Pi Imager, se procede a ejecutar el programa. El cual se debe de visualizar como en la **Figura 5-2**:

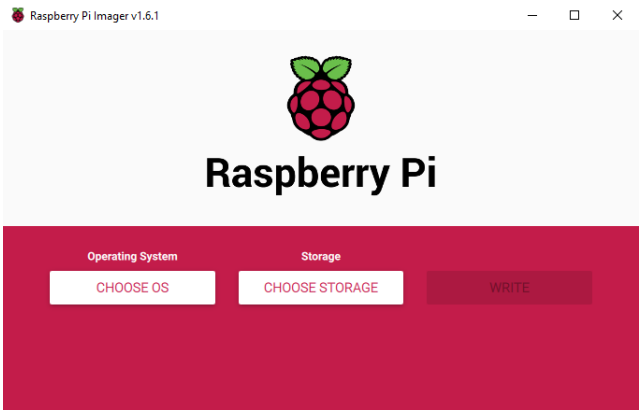


Figura 5-2: Software Raspberry Pi Imager. *[29]*.

En el primer recuadro del programa, se debe de seleccionar el Sistema Operativo que se desea utilizar en la placa Raspberry Pi, hay diferentes opciones: Raspberry Pi OS (con o sin interfaz gráfica), Ubuntu (con imágenes: Desktop, Server o Core), Manjaro ARM Linux (Server o Desktop), RISC OS PI, entre otras opciones de SO.

Al contar con una tarjeta SD de 32 GB se ha seleccionado la opción predeterminada, que es Raspberry Pi OS (32-bit), la cual cuenta con interfaz gráfica. Para el caso de las opciones de almacenamiento aparecerán las posibles configuraciones a seleccionar, en este caso se muestra únicamente la tarjeta SD por lo que se selecciona y se procede a seleccionar el botón **WRITE**, el cual escribirá el SO en la tarjeta.

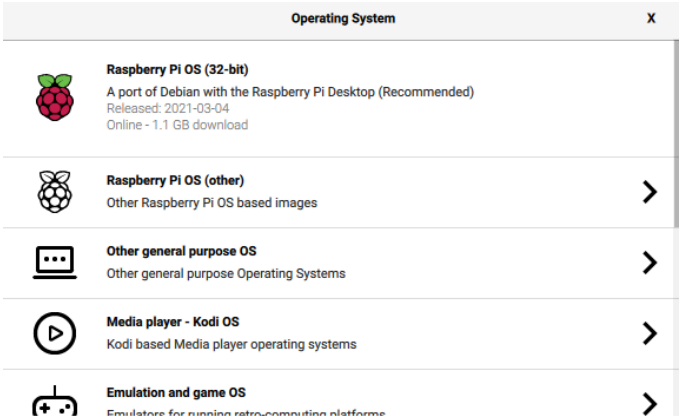


Figura 5-3: Sistemas Operativos para Raspberry Pi. *[29]*.

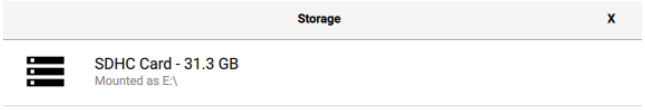


Figura 5-4: Almacenamiento. *[29]*.

Figura 5-5: Resumen de las opciones seleccionadas. *[29]*.

Antes de empezar a escribir en la tarjeta, el programa indicará que todos los datos existentes en la tarjeta SDHC (Secure Digital High Capacity) Card serán eliminados, por lo que preguntará si se desea continuar. Al seleccionar que sí, el programa Raspberry Pi Imager comenzará a escribir la imagen en la tarjeta SD.

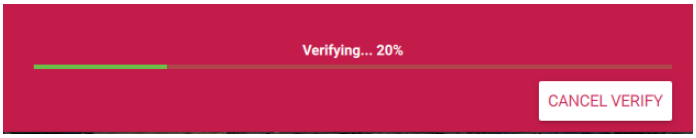


Figura 5-6: Verificación de la tarjeta SD. *[29]*.

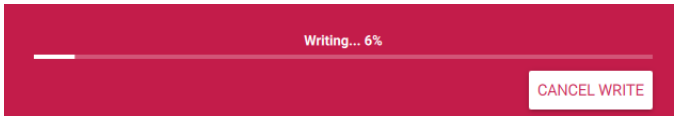


Figura 5-7: Escribiendo el SO en la tarjeta SD. *[29]*.

Al terminar, el programa avisa que el Sistema Operativo seleccionado se ha escrito correctamente por lo que se puede retirar la tarjeta SD del ordenador.

5.3. Conectando la Raspberry Pi.

Al momento de conectar la Raspberry Pi, es importante realizarlo en el orden correcto para que todos los componentes funcionen y estén seguros. Como primer paso se debe insertar la tarjeta SD configurada con el sistema operativo en la ranura para tarjetas SD que se encuentra en la parte inferior de la Raspberry Pi.

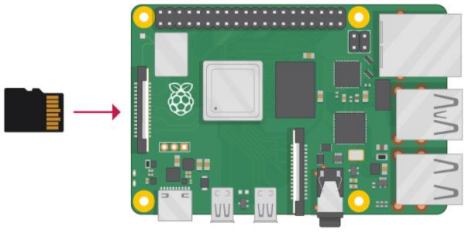


Figura 5-8: Insertar tarjeta SD. *[30]*.

Seguidamente se debe conectar el mouse al puerto USB en la Raspberry Pi, de igual forma se realizará la conexión del teclado en el otro puerto USB.

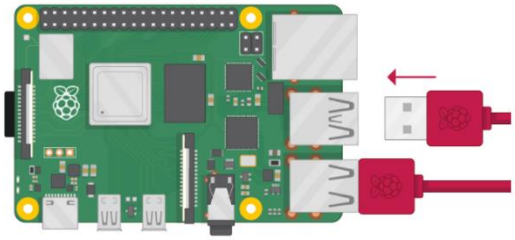


Figura 5-9: Conectar mouse y teclado a puertos USB. *[30]*.

A continuación, se procederá a conectar el monitor que se utilizará como pantalla con el cable HDMI, mientras que el otro extremo del cable se conecta con el puerto HDMI de la Raspberry Pi.

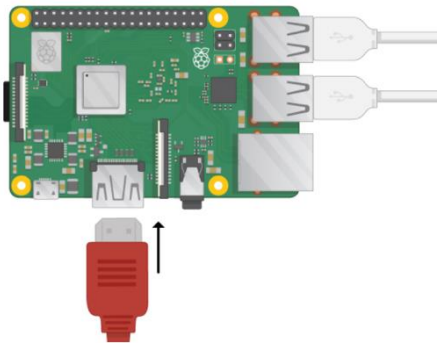


Figura 5-10: Conectar HDMI a la Raspberry. *[30]*.

Para continuar, se requiere conectar la placa a Internet, para ello se usa el cable Ethernet, el cual se conecta al puerto Ethernet de la Raspberry Pi y el otro extremo del cable a la toma que se encuentra en el router.

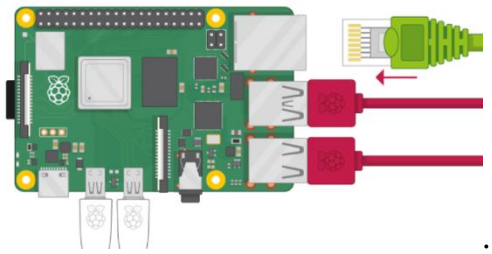


Figura 5-11: Conectar cable Ethernet a la Raspberry Pi. *[30]***.**

Una vez conectados todos los elementos a la Raspberry Pi correctamente, se debe proceder a enchufar la fuente de alimentación y conectarla al puerto de alimentación de la placa.

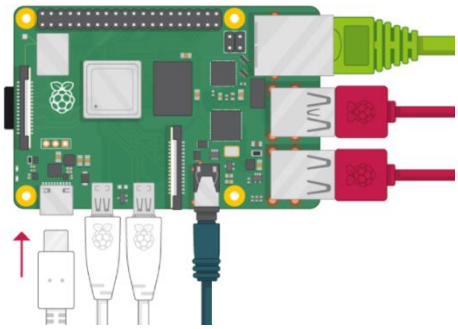


Figura 5-12: Conectar la Raspberry Pi a la Fuente de Alimentación. *[30]***.**

Al conectarse se puede observar que se enciende una luz LED en la Raspberry Pi, lo que indica que la placa se encuentra conectada a la alimentación. Después de unos segundos, en donde el sistema operativo se reinicia automáticamente, la Raspberry solicitará ingresar el usuario y la contraseña por defecto de la Raspberry Pi, los cuales son: **pi** (usuario), **raspberry** (contraseña) y de esta manera aparecerá el escritorio del SO (Sistema Operativo) de la Raspberry Pi.

Figura 5-13: Escritorio Raspberry Pi.

Seguidamente aparecerá la aplicación de Bienvenida de Raspberry Pi, donde se realiza la configuración del sistema. Haciendo clic en *Siguiente*, el programa solicitará configurar el país, el idioma y la zona horaria a usar; donde se selecciona España, español y a Madrid como zona horaria.

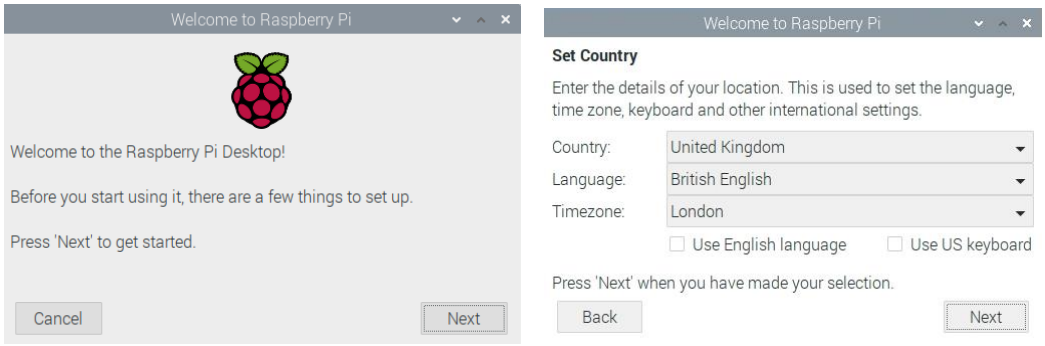


Figura 5-14: Bienvenida y Configuración del sistema.

Se debe de ingresar una nueva contraseña para la Raspberry Pi y el sistema continuará con el proceso, solicitando la actualización del sistema operativo.

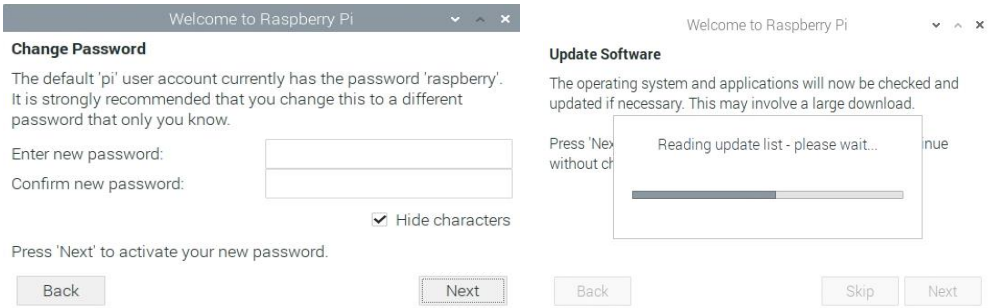


Figura 5-15: Cambio de contraseña y actualización del SO.

Para finalizar la configuración, el programa solicitará reiniciar el sistema, de esta forma se podrá completar la actualización. Finalmente, se puede hacer uso del sistema operativo de la Raspberry Pi por lo que se procederá a realizar la configuración de la placa para poder disfrutar del servicio Amazon Rekognition.

5.4. Instalación de paquetes básicos y del SDK de Rekognition.

En este punto se ha realizado la instalación de los paquetes básicos necesarios y del SDK de Python para poder hacer uso de Amazon Rekognition en la Raspberry Pi. Para ello se deben seguir los pasos explicados en el punto **3.4**, es decir, donde se realiza la actualización de los paquetes disponibles en la máquina, se instala el paquete *aptitude* que permite mejorar la experiencia de instalar los paquetes y finalmente se ha instalado los paquetes básicos con el comando: **sudo aptitude install build-essential**.

Objetivo del comando	Comandos
Actualizar los paquetes disponibles en la máquina	sudo apt-get update && sudo apt-get upgrade
Instalar paquete aptitude.	sudo apt-get install aptitude
Instalar los paquetes básicos.	sudo aptitude install build-essential

Tabla 5-1: Resumen de comandos para instalar paquetes básicos Ubuntu.

Como primer paso para instalar el programa de Python se debe de actualizar los paquetes del sistema, una vez actualizados se procede a instalar mediante el siguiente comando: **sudo apt install python3**. La instalación del administrador de paquetes de Python3 (**pip3**) se realiza con este comando: **sudo apt update && sudo apt install python3-pip**. Para confirmar la instalación se Python se ha utilizado el comando: **python3 -V**, como se muestra en la **Figura 5-16**.

Objetivo del comando	Comandos
Actualizar los paquetes disponibles en la máquina	sudo apt-get update && sudo apt-get upgrade
Instalar Python3.	sudo apt install python3
Actualizar los paquetes e instalar pip3.	sudo apt update && sudo apt install python3-pip
Verificar la versión de Python instalada	python3 -V

Tabla 5-2: Resumen de comandos para instalar Python y pip3.

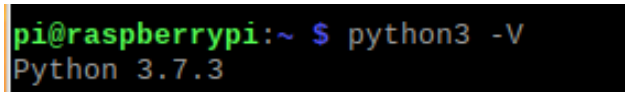


Figura 5-16: Versión de Python instalada.

En cuanto a la instalación del SDK (Boto3) se hizo uso del administrador de paquetes de Python mediante el comando: **pip3 install boto3**. Una vez descargado el SDK, se procede a realizar la instalación de la línea de comandos de Amazon (AWS CLI) en la Raspberry Pi, para ello se debe hacer uso del siguiente comando: **pip3 install awscli --upgrade –user**. Para continuar, se agrega el ejecutable de la AWS CLI a la ruta de línea de comandos de la siguiente manera: **export PATH=/home/pi/.local/bin:$PATH**. Finalmente se procede a confirmar la instalación: **aws --version**.

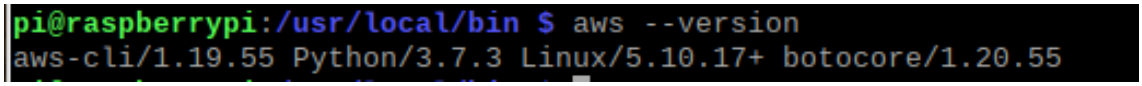


Figura 5-17: Versión de AWS CLI instalada.

Objetivo del comando	Comandos
Instalar el SDK de Python.	pip3 install boto3
Instalar el AWS CLI.	pip3 install awscli --upgrade –user
Añadir el ejecutable de la AWS CLI a la ruta de línea de comandos.	export PATH=/home/pi/.local/bin:$PATH
Verificar la versión de AWS CLI instalada.	aws --version

Tabla 5-3: Resumen de comandos para instalar SDK en Raspberry.

Con los comandos AWS CLI, se ha configurado un perfil llamado **ricardo**, mediante el comando: **aws configure --profile ricardo**, al cual se le han otorgado las credenciales del usuario que se ha creado en Amazon IAM. De igual forma se le asigna la región que se ha seleccionado previamente para ejecutar Amazon Rekognition y se indica JSON como formato de salida.

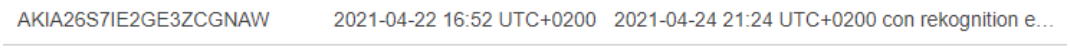


Figura 5-18: Credenciales de usuario IAM para la Raspberry Pi.

Al hacer uso de los programas de Python que se han creado anteriormente, el programa generaba una advertencia debido a que los paquetes *urllib3* y *chardet* no eran soportados por la versión Python que se tenía instalada en la Raspberry. Para solucionar este problema se ha ejecutado el comando: **pip install --upgrade requests**, este comando lo que hace es actualizar manualmente el paquete de solicitudes haciendo uso del administrador de paquetes de Python (**pip**).

Objetivo del comando	Comandos
Crear mediante AWS CLI un perfil de usuario.	aws configure --profile ricardo
Actualizar manualmente el paquete de solicitudes.	pip install --upgrade requests

Tabla 5-4: Resumen de comandos para la configuración de la Raspberry.

5.5. Conexión de la Raspberry Pi por SSH.

Al realizar la configuración de la tarjeta Raspberry Pi no resultaría necesario utilizar los periféricos: ratón, teclado y monitor, pudiendo conectarse al nodo IoT de forma remota por Ethernet.

SSH se encuentra desactivado por defecto en las Raspberry Pi para evitar que el acceso externo sea demasiado fácil. Existen 4 maneras que permiten realizar la activación de SSH, las cuales se explicarán a continuación:

- **Crear archivo SSH en el directorio de la tarjeta SD**: Si no es posible acceder a la placa con los periféricos, una forma de activar SSH es accediendo a la tarjeta SD desde un ordenador externo y crear un archivo llamado *ssh* en el directorio de arranque.
- **Activación del servidor SSH en el escritorio**: Si se tiene acceso a la Raspberry con los periféricos y haciendo uso de la interfaz de escritorio. Se puede acceder al menú de Inicio, se ha seleccionado Preferencias y del submenú se selecciona Configuración de Raspberry Pi, en la pestaña Interfaces de los ajustes de la placa se encuentra la entrada para SSH y se procede a activarla.
- **Habilitar SSH por la terminal en raspi-config**: En la línea de comandos, mediante el comando **sudo raspi-config**, se activará el menú de la herramienta de configuración, se selecciona la opción 7 ("Operaciones avanzadas") y después la opción A4 ("SSH"). Para finalizar, se debe aceptar la configuración para activar el servidor SSH.
- **Iniciar el servicio SSH con systemctl**: De igual forma, mediante la línea de comandos systemctl se puede configurar la Raspberry PI, introduciendo estos dos comandos: **sudo systemctl enable ssh** y **sudo systemctl start ssh**. Para finalizar, se debe de reiniciar la placa con el comando **sudo reboot**.

Para poder acceder de forma remota a la Raspberry Pi, es necesario conocer la dirección IP de la placa, esto se obtiene gracias al comando: **hostname -I**. Mediante el software PuTTY en Windows, se puede realizar la conexión remota con la placa, para ello se indica en el Host Name o la dirección IP de la Raspberry y el puerto número 22; al realizar la conexión se visualiza un mensaje solicitando confirmar la fiabilidad del servidor SSH de la Raspberry Pi y la clave SSH.

La configuración se habrá realizado de forma correcta cuando se visualice el nombre del usuario, en este caso **pi** y como dirección IP **raspberrypi,** lo que indica que la conexión funciona a la perfección, como visualiza en la **Figura 5-19**.

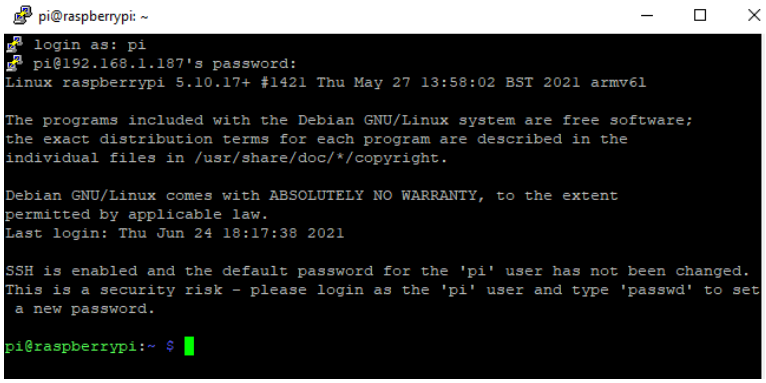


Figura 5-19: Raspberry Pi conectada por SSH.

5.6. Reconocimiento de etiquetas personalizadas en la Raspberry Pi.

En este apartado se plantea hacer uso de las etiquetas personalizadas para identificar objetos con características similares, pero que el algoritmo sea capaz de encontrar las diferencias a la hora de reconocer cada objeto. Es decir, se ha comprobado que el sistema reconoce sin problema cuando en la imagen hay un Libro, sin embargo, se plantea que el algoritmo de reconocimiento sea capaz de identificar cada uno de los libros que aparecen en la imagen.

Para llevar a cabo el reconocimiento de Libros es necesario crear un nuevo dataset, debido a que el que se había creado con anterioridad cuenta con las imágenes de libros y llaves, por lo tanto, se ha llamado al conjunto de datos **Identifica-libros**. También se cuenta con el bucket **demo-libros-llaves**, en donde se almacenan todas las imágenes que se han ido utilizando, por lo que se ha creado dentro del bucket una nueva carpeta llamada **Libros a identificar**, en dicha carpeta se han incluido diferentes imágenes de 5 libros que se desean reconocer.

Con el dataset creado se debe proceder a realizar el reconocimiento de las etiquetas en cada una de las imágenes. Una vez identificadas las 5 etiquetas se ha procedido a realizar el entrenamiento del modelo bajo la opción de 80% entrenamiento y 20% pruebas, se ha esperado 1 hora para que el modelo termine de entrenarse y de esta forma se encuentre listo para su uso. Finalizado el procedimiento, se han obtenido las siguientes puntuaciones de precisión y recuperación para cada una de las etiquetas:

Label name	F1 score	Test images	Precision	Recall	Assumed threshold
Apuntes sobre conceptos básicos para la iniciación en Econometría	1.000	3	1.000	1.000	0.200
El arte de vivir con sencillez	1.000	2	1.000	1.000	1.000
El libro que tu cerebro no quiere leer	1.000	2	1.000	1.000	1.000
La aventura de Diana	1.000	3	1.000	1.000	1.000
Libera tu magia	1.000	2	1.000	1.000	1.000

Figura 5-20: Precisión y Recuperación de los libros a identificar.

Con los datos analizados se identifica que el modelo ha logrado obtener una puntuación de precisión y recuperación perfecta, lo que indica que el modelo de reconocimiento realizará las predicciones correctamente de cada libro y no habrá perdida en cuanto a predicciones reales, logrando de esta manera el modelo idóneo, porque el nivel de precisión al realizar la identificación es el ideal.

Para poder realizar la detección de objetos en la Raspberry Pi es necesario contar con 3 programas, el primero se ha llamado **istart.py** (descargado de Amazon), el cual contiene la información del modelo para poder iniciarlo y así poder utilizar las etiquetas. El segundo programa se llama **icustom.py**, en el cual se indica a la API de Rekognition la imagen que se desea analizar y esta devolverá el nombre del libro que ha sido identificado. Para finalizar, el último programa se ha llamado **istop.py**, el cual ha sido proporcionado por Amazon para detener el modelo una vez se haya terminado de utilizar. El código de los programas se encuentra en el **Anexo I**, apartados **A.9**, **A.10** y **A.11**.

Con dichos programas configurados correctamente, se procederá a realizar la identificación de los libros, para de esta forma poder comprobar que el análisis que realiza el algoritmo de reconocimiento funciona correctamente.

Libro 1: Libera tu Magia.

Figura 5-21: Imagen del libro.

```
pi@raspberrypi:~/Python $ python3 icustom.py
El libro que se ha identificado en la imagen es:
Libera tu magia , con un porcentaje del:  100.0 %
```

Figura 5-22: Reconocimiento del libro por el programa icustom.py.

Como se puede observar en la **Figura 5-22**, el libro *Libera tu magia* ha sido reconocido por el modelo.

Libro 2: La aventura de Diana.

Figura 5-23: Imagen del libro.

```
pi@raspberrypi:~/Python $ python3 icustom.py
El libro que se ha identificado en la imagen es:
La aventura de Diana , con un porcentaje del:  100.0 %
```

Figura 5-24: Reconocimiento del libro por el programa icustom.py.

En esta prueba, el programa ha identificado correctamente que el libro que se deseaba detectar era: *La aventura de Diana*.

Libro 3: Apuntes sobre conceptos básicos para la iniciación en Econometría.

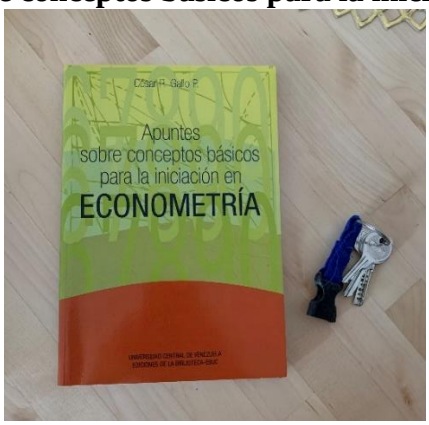


Figura 5-25: Imagen del libro.

```
pi@raspberrypi:~/Python $ python3 icustom.py
El libro que se ha identificado en la imagen es:
Apuntes sobre conceptos básicos para la iniciación en Econometría , con un porcentaje del:  100.0 %
```

Figura 5-26: Reconocimiento del libro por el programa icustom.py.

Para realizar la demostración del modelo se han utilizado 3 de los 5 libros que han sido etiquetados. De igual forma, el proceso de identificación de las etiquetas se ha realizado para todas, teniendo un éxito del 100%, al detectar correctamente los 5 libros.

Este estudio lo que permite identificar es que Rekognition es capaz de diferenciar entre objetos con características muy similares, y realizarlo con un alto porcentaje de precisión, siempre y cuando se cuente con una buena base de datos para entrenar el modelo.

5.7. Configuración y reconocimiento de objetos con Cámara Raspberry Pi.

Para finalizar este capítulo se plantea integrar a la Raspberry Pi el módulo de Cámara Pi, como el que se muestra en la **Figura 5-26**, el cual permitirá tomar fotos en tiempo real para que Amazon Rekognition analice las imágenes y que de esta forma se puedan detectar las etiquetas personalizadas.

Figura 5-27: Cámara Raspberry Pi. *[31]***.**

Una vez conectado el cable de cinta flexible de la cámara a la placa y asegurando que los conectores del cable están bien conectados, se procederá a realizar los ajustes de la cámara, para ello se abre la herramienta de configuración de Raspberry, accediendo desde el menú principal, seleccionando las **Preferencias** y del submenú la opción **Configuración de la Raspberry**. Para finalizar se abre la ventana *Interface* y en ella se habilita la cámara, de esta manera la configuración se ha realizado correctamente.

Para poder habilitar la cámara con Python, es necesario hacer uso del módulo *picamera*, por lo que se ha importado dicho módulo al programa **icustom.py,** de igual forma el contenido se debe de modificar para poder tomar fotos. Con la foto capturada, es necesario enviar la imagen al bucket **demo-libros-llaves** para que el algoritmo de reconocimiento sea capaz de detectar los objetos que aparecen en la foto. El contenido del programa actualizado se encuentra en el **Anexo I,** apartado **A.12.**

Con la nueva configuración del programa se han capturado 2 imágenes de los libros etiquetados para que de esta forma el modelo sea capaz de reconocerlos. En la

Figuras 5-29 y **5-31**, se puede observar que el sistema ha detectado los libros correctamente.

Libro 1: El arte de vivir con sencillez.

Figura 5-28: Imagen del libro.

```
pi@raspberrypi:~/Python $ python3 icustom.py
El libro que se ha identificado en la imagen es:
El arte de vivir con sencillez , con un porcentaje del:  100.0 %
```

Figura 5-29: Reconocimiento del libro por el programa icustom.py.

Libro 2: El libro que tú cerebro no quiere leer.

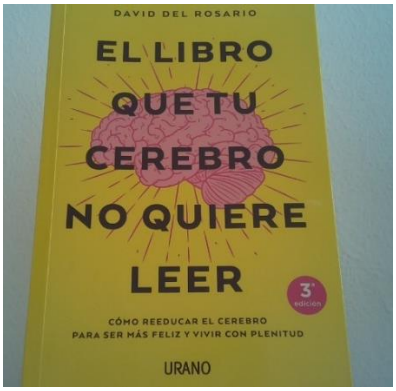


Figura 5-30: Imagen del libro.

```
pi@raspberrypi:~/Python $ python3 icustom.py
El libro que se ha identificado en la imagen es:
El libro que tu cerebro no quiere leer , con un porcentaje del:  100.0 %
```

Figura 5-31: Reconocimiento del libro por el programa icustom.py.

Esta configuración demuestra que Amazon Rekognition es capaz de detectar las etiquetas personalizadas, con un alto nivel de precisión, en imágenes capturadas en tiempo real, siempre y cuando estás se guarden en un bucket de Amazon S3. Esto abre una línea de trabajo futuro que se comentará en el próximo capítulo.

6. Conclusiones y trabajo futuro.

En este punto se resumen las conclusiones de cada uno de los capítulos anteriores, permitiendo comprobar que los objetivos del TFG se han logrado cumplir satisfactoriamente. Para finalizar, se comentará una línea de trabajo futuro con la aplicación de IoT Greengrass.

6.1. Conclusiones del TFG.

La finalidad de este Trabajo de Fin de Grado era la realizar un análisis de los pasos a seguir para hacer uso de un servicio de Inteligencia Artificial en la nube. Para comprender la utilidad de Amazon Rekognition frente a otros sistemas, se ha realizado una comparación que permitió conocer que Rekognition y Google Cloud Vision API son las herramientas más potentes y precisas a la hora de reconocer objetos en una imagen. Sin embargo, la ventaja de Amazon Rekognition es que se ha especializado en el desarrollo de etiquetas personalizadas, lo que permite a cualquier persona realizar una detección automatizada mediante el entrenamiento del algoritmo, de esta forma el sistema será capaz de detectar objetos particulares.

Para poder comprender las diferentes utilidades que se le puede dar al uso del servicio de IA de AWS, se ha empezado por crear una Instancia Virtual en la que se deben de realizar diferentes configuraciones tales como instalación de paquetes básicos e instalación del SDK, lo que permitirá hacer uso de Rekognition. Para poder utilizar cualquier servicio de AWS, es necesario crear un usuario IAM, al cual se le deben de otorgar los diferentes permisos de usuario que permitan poder hacer uso del SDK de Rekognition.

Con la configuración de la máquina virtual, se ha procedido a estudiar las diferentes detecciones de objetos que permite realizar Amazon Rekognition. En primer lugar, se ha diseñado un programa que realiza la detección de objetos de una imagen que se encuentre disponible en Internet, este algoritmo ha sido entrenado por Amazon, por lo que realiza el reconocimiento de todos los objetos de la imagen, obteniendo una larga lista de ítems detectados en las diferentes pruebas que se realizaron. Dicho algoritmo resulta muy útil de utilizar cuando se quiere tener una visión global de los objetos que se encuentran en la imagen y de igual forma permite comprobar el alto nivel de precisión que tiene el servicio.

Resulta curioso poder hacer pruebas con objetos personalizados, para ello es necesario crear etiquetas customizadas. Esto permite aportar todos los datos de entrenamiento de las diferentes etiquetas que se desean reconocer (mientras más datos se

aporten, más porcentaje de precisión tendrá el modelo). Con las imágenes etiquetadas se procedió a realizar el entrenamiento del algoritmo, es interesante ver como en una hora (dependiendo de la cantidad de datos aportados), el modelo realiza el entrenamiento y las pruebas necesarias para poder notificar el porcentaje de precisión y recuperación que tiene, lo que permite comprobar si el modelo necesita que se aporten más datos para poder mejorar su rendimiento.

Al utilizar algoritmos con etiquetas personalizadas, Amazon crea dos programas que permiten la activación y detención del modelo debido a que es un servicio que se cobra por horas de uso, por lo tanto, se recomienda activar el algoritmo cuando se vaya a utilizar. Con la configuración realizada se ha comprobado que el modelo puede reconocer objetos particulares, generando una gran cantidad de usos para este servicio, tanto para proyectos particulares como para diferentes áreas de trabajo como: industrias, sector salud, empresas de entretenimiento, etc.

Como la configuración inicial del modelo se realizó en un ordenador virtual con conexión a Internet, se quiso comprobar que los algoritmos de reconocimiento también funcionan en sistemas de bajo consumo, con baja potencia y conectados a la red como el caso de los nodos IoT. Para ello se ha implementado en una Raspberry Pi, en la cual es necesario realizar la configuración del sistema operativo y la descarga de los paquetes que esenciales y del SDK, pero aplicados al entorno de la Raspberry.

Para los nodos IoT se ha querido realizar el reconocimiento de objetos del mismo tipo, pero con algunas características diferentes que permitiesen comprobar el nivel de precisión al que el algoritmo era capaz de llegar. Para ello, se ha creado un nuevo modelo que permitiese detectar que libro (de los 5 etiquetados), es el que aparece en la imagen y al contar con una buena base de datos se ha podido comprobar que el modelo es capaz de diferenciar cada uno de los libros con un porcentaje de precisión muy alto. Esto nos deja como reflexión que el uso de etiquetas personalizadas no solo puede servir para identificar objetos completamente diferentes, sino que también permite diferenciar entre objetos del mismo tipo, por lo que se puede aplicar en sistemas de clasificación de objetos, aumentando de esta forma el nivel de precisión de los mismo.

6.2. Línea de trabajo futuro.

Como se ha mencionado al final del capítulo 5, la integración de la Raspberry Pi con la cámara crea una línea de trabajo futuro, al poderse implementar con el servicio AWS IoT Greengrass. Este servicio permite a los dispositivos conectados ejecutar funciones de AWS Lambda, contenedores Docker o ambos, realizar predicciones basadas en modelos de aprendizaje automático, mantener los datos de dispositivos sincronizados y comunicarse con otros dispositivos de forma segura sin la necesidad de estar conectados a Internet [32].

Lo que se plantea como trabajo futuro es configurar la placa Raspberry Pi para que ejecute AWS IoT Greengrass Core y que le permita de esta forma, comunicarse con otros dispositivos de AWS IoT que tengan instalado el SDK como, por ejemplo, Echo Dot (altavoz inteligente de Amazon que usa el servicio de voz conocido como Alexa) o con otras Raspberrys, para realizar una configuración más compleja.

AWS Lambda es un servicio informático sin servidor que permite ejecutar código sin aprovisionar ni administrar servidores, mantener integraciones de eventos o administrar tiempos de ejecución. Lambda permite cargar el código mediante un archivo ZIP o una imagen de contenedor, por lo que asigna de manera automática y precisa la potencia de ejecución informática y ejecuta el código en función de la solicitud o un evento entrante para cualquier escala de tráfico.

Como se observa en la **Figura 6-1**, se observa como la placa Raspberry Pi, se puede integrar mediante las funciones Lambda con los servicios Cloud de AWS gracias al uso de IoT Greengrass.

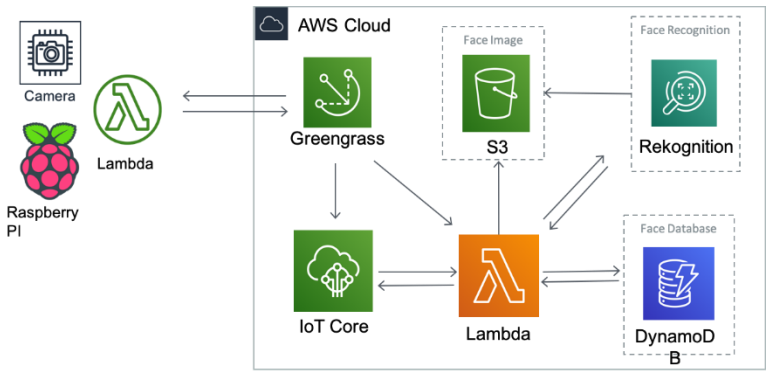


Figura 6-1: Integración de Raspberry Pi con IoT Greengrass. *[33].*

Bibliografía

[1] L. Rouhiainen, Inteligencia Artificial. 101 Cosas que debes saber hoy sobre nuestro futuro., Barcelona: Planeta, SA., 2018.

[2] Redacción APD, «APD,» [En línea]. Available: https://www.apd.es/tecnicas-de-la-inteligencia-artificial-cuales-son-y-para-que-se-utilizan/. [Último acceso: 5 Abril 2021].

[3] R. Alonso, «Hard Zone,» 8 Abril 2020. [En línea]. Available: https://hardzone.es/tutoriales/rendimiento/diferencias-ia-deep-machine-learning/. [Último acceso: 6 Abril 2021].

[4] Iberdrola, «Iberdrola,» [En línea]. Available: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico. [Último acceso: 8 Abril 2021].

[5] G. P. Victor, «Telefónica Tech,» 18 Enero 2019. [En línea]. Available: https://empresas.blogthinkbig.com/una-breve-historia-del-machine-learning/. [Último acceso: 8 Abril 2021].

[6] Lawtomated, «Lawtomated,» 28 Abril 2019. [En línea]. Available: https://lawtomated.com/a-i-technical-machine-vs-deep-learning/. [Último acceso: 10 Abril 2021].

[7] SmartPanel, «SmartPanel,» [En línea]. Available: https://www.smartpanel.com/que-es-deep-learning/. [Último acceso: 10 Abril 2021].

[8] Contaval, «Contaval,» [En línea]. Available: https://www.contaval.es/que-es-la-vision-artificial-y-para-que-sirve/. [Último acceso: 15 Abril 2021].

[9] El Efete, «El Efete,» [En línea]. Available: http://www.elefete.com/la-inteligencia-artificial-en-la-nube-el-nuevo-modelo-de-negocio-de-las-gigantes-tecnologicas/. [Último acceso: 15 Junio 2021].

[10] Barbara IoT, «Barbara,» [En línea]. Available: https://barbaraiot.com/blog/nodos-iot/. [Último acceso: 15 Junio 2021].

[11] Amazon, «Rekogntion,» [En línea]. Available: https://aws.amazon.com/es/rekognition/?blog-cards.sort-by=item.additionalFields.createdDate&blog-cards.sort-order=desc. [Último acceso: 4 Marzo 2021].

[12] Amazon, «Amazon Rekognition Image,» [En línea]. Available: https://aws.amazon.com/es/rekognition/image-features/?nc=sn&loc=3&dn=2. [Último acceso: 24 Abril 2021].

[13] Amazon, «Amazon Rekognition Video,» [En línea]. Available: https://aws.amazon.com/es/rekognition/image-features/?nc=sn&loc=3&dn=2. [Último acceso: 24 Abril 2021].

[14] Microsoft Azure, «Cognitive Services,» [En línea]. Available: https://azure.microsoft.com/es-es/services/cognitive-services/. [Último acceso: 26 Abril 2021].

[15] R. Ranjan, «Analytics Vidhya,» [En línea]. Available: https://www.analyticsvidhya.com/blog/2020/12/microsoft-azure-cognitive-services-api-for-ai-development/.

[16] IBM, «IBM Cloud Visual Recognition,» [En línea]. Available: https://cloud.ibm.com/docs/visual-recognition?topic=visual-recognition-index. [Último acceso: 28 Abril 2021].

[17] Google Cloud, «Vision AI,» [En línea]. Available: https://cloud.google.com/vision. [Último acceso: 28 Abril 2021].

[18] SourceForge, «SourceForge,» [En línea]. Available: https://sourceforge.net/software/compare/Amazon-Rekognition-vs-Azure-Computer-Vision-vs-Google-Cloud-Vision-API-vs-Watson-Visual-Recognition/. [Último acceso: 28 Mayo 2021].

[19] ActiveWizards, «ActiveWizards,» [En línea]. Available: https://activewizards.com/blog/comparison-of-the-top-cloud-apis-for-computer-vision/. [Último acceso: 28 Mayo 2021].

[20] E. Brown, «ZDNet,» [En línea]. Available: https://www.zdnet.com/article/which-company-does-the-best-job-at-image-recognition-microsoft-amazon-google-or-ibm/. [Último acceso: 28 Mayo 2021].

[21] PuTTY, «Download PuTTY,» [En línea]. Available: https://www.putty.org/. [Último acceso: 2 Abril 2021].

[22] Amazon , «Recursos de Amazon Rekognition,» [En línea]. Available: https://aws.amazon.com/es/rekognition/resources/. [Último acceso: 3 Abril 2021].

[23] Amazon, «AWS IAM,» [En línea]. Available: https://aws.amazon.com/es/rekognition/resources/. [Último acceso: 6 Abril 2021].

[24] Boto3., «Boto3 Documentation,» [En línea]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rekognition.html. [Último acceso: 5 Abril 2021].

[25] Autocasión., «Autocasión.,» [En línea]. Available: https://www.autocasion.com/actualidad/noticias/en-esta-calle-de-londres-cada-farola-recarga-coches-electricos. [Último acceso: 7 Abril 2021].

[26] El Universo, «El universo,» [En línea]. Available: https://www.eluniverso.com/noticias/2020/05/28/nota/7854669/semaforo-amarillo-rojo-verde-guayaquil-quito-cuenca-ecuador-covid/. [Último acceso: 6 Abril 2021].

[27] Detectfire, «Detectfire & Security,» [En línea]. Available: http://detecfire.com/wp-content/uploads/2017/01/hidrante-contra-incendios.jpg. [Último acceso: 6 Abril 2021].

[28] ELLE Decor, «ELLE Decor,» [En línea]. Available: https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/foto-escritorio-1524999265.jpg?resize=480:*. [Último acceso: 6 Abril 2021].

[29] Raspberry Pi, «Raspberry Pi OS,» [En línea]. Available: https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/foto-escritorio-1524999265.jpg?resize=480:*. [Último acceso: 16 Abril 2021].

[30] Raspberry Pi, «Conectando con Raspberry Pi,» [En línea]. Available: https://projects.raspberrypi.org/es-ES/projects/raspberry-pi-getting-started/3. [Último acceso: 26 Abril 2021].

[31] ArduCam, «Arducam,» [En línea]. Available: https://www.arducam.com/product/b0033c-arducam-5mp-1080p-camera-module-with-case-for-raspberry-pi-3-3-b-and-more/. [Último acceso: 21 06 2021].

[32] Amazon, «AWS IoT Greengrass,» [En línea]. Available: https://aws.amazon.com/es/greengrass/. [Último acceso: 1 Junio 2021].

[33] ECloudture, «ECloudture,» [En línea]. Available: https://www.ecloudture.com/en/aiot-face-recognition-with-raspberry-pi-2/. [Último acceso: 10 Junio 2021].

Lista de Acrónimos y Abreviaturas.

AMI	Imágenes de máquina de Amazon
API	Application Programming Interface
ARN	Amazon Resources Name
AWS	Amazon Web Services.
CLI	Command Line Interface.
CPU	Central Processing Unit
DL	Deep Learning.
DNS	Domain Name System
GB	Gigabyte
HDMI	High-Definition Multimedia Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
IA	Inteligencia Artificial.
IAM	Identity and Access Management
IP	Internet Protocol
JSON	JavaScript Object Notation
JPEG	Joint Photographic Experts Group
ML	Machine Learning.
PIP	Paquetes de Instalación PIP
PNG	Portable Network Graphics
SaaS	Software as a Service
SD	Secure Digital
SDHC	Secure Digital High Capacity
SDK	Software Development Kits
SO	Sistema Operativo
SSH	Secure Shell
S3	Simple Storage Service
TCP	Transmission Control Protocol
TFG	Trabajo Final de Grado.
UAH	Universidad de Alcalá.
URL	Uniform Resource Locator
USB	Universal Serial Bus

A. Anexo I – Explicación códigos de programas de Python.

En este anexo, se encuentran todos los códigos de los diversos programas que se han desarrollado a lo largo del Trabajo de Fin de Grado.

A.1. Image.py.

Este programa se utiliza para la transformación de la URL de la imagen en formato bytes. Como se requiere enviar solicitudes HTTP mediante Python, es necesario importar en el programa el módulo *requests*. La solicitud HTTP devuelve un Response Object que contiene todos los datos de respuesta (contenido, codificación, estado, etc).

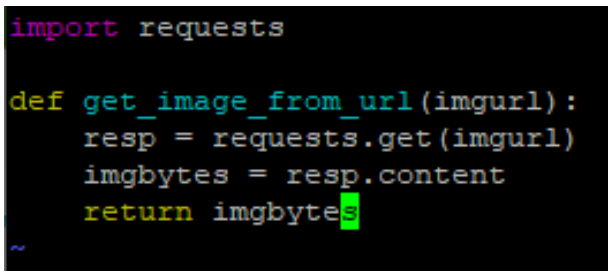


Figura A-1: Código del programa image.py.

La función **get_image_from_url**, recibe como parámetro la URL de la imagen, después se desea obtener el Objeto Response (llamado en el programa **resp**), para ello se usa la función *get* del módulo *requests*, con dicho objeto se puede obtener toda la información necesaria.

Mediante la función *content*, se puede acceder al cuerpo del Objeto Response, obteniendo de esta forma la imagen en bytes, lo que finalmente se utilizará por Rekognition para el análisis de objetos; por lo tanto, el dato **imgbytes** es el parámetro que el programa devolverá.

A.2. Demo.py.

Este programa hace uso del SDK de Rekognition para llamar a la API y obtener de esta forma la lista de objetos que han sido identificados. Para ello es necesario importar el módulo *boto3* (SDK) e *image* (programa Python creado anteriormente).

```
#Esto es una demo de AWS Rekognition
import boto3
import image

client = boto3.client('rekognition')
imgurl = 'https://images1.autocasion.com/unsafe/1200x800/actualidad/wp-content/uploads/
2020/03/electric-avenue.jpg'
imgbytes = image.get_image_from_url(imgurl)
rekresp = client.detect_labels(Image={'Bytes': imgbytes})
#pprint(rekresp)
print("Esto es lo que observamos de la imagen:")
for label in rekresp['Labels']:
    print(label['Name'],"con un porcentaje del: ",label['Confidence'],"%")
~
```

Figura A-2: Código del programa demo.py.

La función *client* del módulo *boto3*, permite crear un cliente de bajo nivel, en este caso representa al servicio Amazon Rekognition ***('rekognition')***, haciendo referencia a que se utilizará dicha API. Con la URL de la imagen que se desea analizar se llama a la función *get_image_from_url* del programa **image.py**, procesa la imagen y la devuelve en bytes (**imgbytes**).

Con el cliente que se ha creado, se puede hacer uso de la función *detect_labels*, la cual enviará a la API de Rekognition, la imagen en bytes para que se detecten los objetos de la imagen (**rekresp**).

La respuesta anterior se obtiene en formato JSON lo que permitirá, mediante el uso de un bucle *for*, recorrer la respuesta buscando las diferentes etiquetas que se hayan recibido para de esta forma mostrarlas por Shell con el respectivo nivel de confianza.

A.3. Start.py.

Este programa es proporcionado por Amazon Rekognition, su ejecución permite iniciar el modelo de aprendizaje. Hace uso del módulo *boto3* de Python, se inicia el modelo con el ARN que la función *start_model* recibe como parámetro, el programa espera a que el estado del modelo cambie a ejecutándose, al terminar notifica mediante Shell que se ha realizado correctamente.

```
import boto3

def start_model(project_arn, model_arn, version_name, min_inference_units):
    session=boto3.Session(profile_name='ricardoTFG')
    client=session.client('rekognition')

    try:
        # Start the model
        print('Starting model: ' + model_arn)
        response=client.start_project_version(ProjectVersionArn=model_arn, MinInferenceUnits=mi
n_inference_units)
        # Wait for the model to be in the running state
        project_version_running_waiter = client.get_waiter('project_version_running')
        project_version_running_waiter.wait(ProjectArn=project_arn, VersionNames=[version_name]
)

        #Get the running status
        describe_response=client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])
    except Exception as e:
        print(e)

    print('Done...')

def main():
    project_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-project/1617448142840'
    model_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-project/version/demo-pro
ject.2021-04-03T13.22.26/1617448945639'
    min_inference_units=1
    version_name='demo-project.2021-04-03T13.22.26'
    start_model(project_arn, model_arn, version_name, min_inference_units)

if __name__ == "__main__":
    main()
```

Figura A-3: Código del programa start.py.

A.4. Stop.py.

Este programa es proporcionado por Amazon Rekognition, su ejecución permite detener el modelo de aprendizaje. Hace uso de los módulos *boto3* y *time* de Python, se detiene el modelo con el ARN que la función *stop_model* recibe como parámetro, el programa cambia el estado del modelo cambie a deteniendo, al terminar notifica mediante Shell que se ha realizado correctamente.

```
import image
import boto3
from time import time
#from pprint import pprint
#Custom labels
start_time = time()
model='arn:aws:rekognition:us-east-2:752892782220:project/demo-project/version/demo-project.202
1-04-03T13.22.26/1617448945639'
bucket='demo-libros-llaves'
photo='IMG_4822[1].JPG'

session = boto3.Session(profile_name='ricardoTFG')
client = session.client('rekognition')
rekresp = client.detect_custom_labels(ProjectVersionArn=model, Image={'S3Object':{'Bucket':buck
et,'Name':photo}})
#pprint(rekresp)
print("Esto es lo que observamos de la imagen:")
for label in rekresp['CustomLabels']:
    print(label['Name'],"con un porcentaje del: ",label['Confidence'],"%")

elapsed_time = time() - start_time
print("Tiempo de ejecución: %0.10f segundos." % elapsed_time)
```

Figura A-4: Código del programa stop.py

A.5. Custom.py.

El programa **custom.py**, importa el módulo *boto3* de Python. Se copia el ARN del modelo, el bucket donde se encuentra el dataset de imágenes y la foto que se desea analizar.

Mediante la función *client* del módulo *boto3*, permite indicar que se desea usar el servicio Amazon Rekognition ***('rekognition')***, haciendo referencia a que se utilizará dicha API. La función *detect_custom_labels* pasa como parámetro el ARN del modelo y la imagen en bytes del bucket S3. Para finalizar mediante un bucle *for*, se muestran todas las etiquetas detectadas por Rekognition.

```
import boto3
#from pprint import pprint
#Custom labels
model='arn:aws:rekognition:us-east-2:752892782220:project/demo-project/version/d
emo-project.2021-04-03T13.22.26/1617448945639'
bucket='demo-libros-llaves'
photo='IMG_4822[1].JPG'

client = boto3.client('rekognition')
rekresp = client.detect_custom_labels(ProjectVersionArn=model, Image={'S3Object'
:{'Bucket':bucket,'Name':photo}})
#pprint(rekresp)
print("Esto es lo que observamos de la imagen:")
for label in rekresp['CustomLabels']:
    print(label['Name'],"con un porcentaje del: ",label['Confidence'],"%")
```

Figura A-5: Código del programa custom.py.

A.6. Ostart.py.

Este programa es proporcionado por Amazon Rekognition, su ejecución permite iniciar el modelo de aprendizaje. Hace uso del módulo *boto3* de Python, se inicia el modelo con el ARN que la función *start_model* recibe como parámetro, el programa espera a que el estado del modelo cambie a ejecutándose, al terminar notifica mediante Shell que se ha realizado correctamente.

```python
import boto3

def start_model(project_arn, model_arn, version_name, min_inference_units):
    session=boto3.Session(profile_name='ricardoTFG')
    client=session.client('rekognition')

    try:
        # Start the model
        print('Starting model: ' + model_arn)
        response=client.start_project_version(ProjectVersionArn=model_arn, MinInferenceUnits=mi
n_inference_units)
        # Wait for the model to be in the running state
        project_version_running_waiter = client.get_waiter('project_version_running')
        project_version_running_waiter.wait(ProjectArn=project_arn, VersionNames=[version_name]
)

        #Get the running status
        describe_response=client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])
    except Exception as e:
        print(e)

    print('Done...')

def main():
    project_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-single-object/16175534
65288'
    model_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-single-object/version/de
mo-single-object.2021-04-04T18.31.01/1617553859988'
    min_inference_units=1
    version_name='demo-single-object.2021-04-04T18.31.01'
    start_model(project_arn, model_arn, version_name, min_inference_units)

if __name__ == "__main__":
    main()
```

Figura A-6: Código del programa ostart.py.

A.7. Ostop.py.

Este programa es proporcionado por Amazon Rekognition, su ejecución permite detener el modelo de aprendizaje. Hace uso de los módulos *boto3* y *time* de Python, se detiene el modelo con el ARN que la función *stop_model* recibe como parámetro, el programa cambia el estado del modelo cambie a deteniendo, al terminar notifica mediante Shell que se ha realizado correctamente.

```python
import boto3
import time


def stop_model(model_arn):
    session=boto3.Session(profile_name='ricardoTFG')
    client=session.client('rekognition')

    print('Stopping model:' + model_arn)

    #Stop the model
    try:
        response=client.stop_project_version(ProjectVersionArn=model_arn)
        status=response['Status']
        print ('Status: ' + status)
    except Exception as e:
        print(e)

    print('Done...')

def main():

    model_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-single-object/version/de
mo-single-object.2021-04-04T18.31.01/1617553859988'
    stop_model(model_arn)

if __name__ == "__main__":
    main()
```

Figura A-7: Código del programa ostop.py.

A.8. Ocustom.py.

El programa **ocustom.py**, importa el módulo *boto3* de Python. Se copia el ARN del modelo, el bucket donde se encuentra el dataset de imágenes y la foto que se desea analizar.

Mediante la función *client* del módulo *boto3*, permite indicar que se desea usar el servicio Amazon Rekognition ***('rekognition')***, haciendo referencia a que se utilizará dicha API. La función *detect_custom_labels* pasa como parámetro el ARN del modelo y la imagen en bytes del bucket S3. Para finalizar mediante un bucle *for*, se muestran todas las etiquetas detectadas por Rekognition.

```
import image
import boto3
from time import time
#from pprint import pprint
#Custom labels
start_time=time()
model='arn:aws:rekognition:us-east-2:752892782220:project/demo-single-object/version/demo-singl
e-object.2021-04-04T18.31.01/1617553859988'
bucket='demo-libros-llaves'
photo='IMG_4822[1].JPG'

session = boto3.Session(profile_name='ricardoTFG')
client = session.client('rekognition')
rekresp = client.detect_custom_labels(ProjectVersionArn=model, Image={'S3Object':{'Bucket':buck
et,'Name':photo}})
#pprint(rekresp)
print("Esto es lo que observamos de la imagen:")
for label in rekresp['CustomLabels']:
    print(label['Name'],"con un porcentaje del: ",label['Confidence'],"%")

elapsed_time = time() - start_time
print("Tiempo de ejecución: %0.10f segundos." % elapsed_time)
```

Figura A-8: Código del programa ocustom.py.

A.9. Istart.py.

Este programa es proporcionado por Amazon Rekognition, su ejecución permite iniciar el modelo de aprendizaje. Hace uso del módulo *boto3* de Python, se inicia el modelo con el ARN que la función *start_model* recibe como parámetro, el programa espera a que el estado del modelo cambie a ejecutándose, al terminar notifica mediante Shell que se ha realizado correctamente.

```python
import boto3

def start_model(project_arn, model_arn, version_name, min_inference_units):
    session=boto3.Session(profile_name='ricardo')
    client=session.client('rekognition')

    try:
        # Start the model
        print('Starting model: ' + model_arn)
        response=client.start_project_version(ProjectVersionArn=model_arn, MinInferenceUnits=min_inference_units)
        # Wait for the model to be in the running state
        project_version_running_waiter = client.get_waiter('project_version_running')
        project_version_running_waiter.wait(ProjectArn=project_arn, VersionNames=[version_name])

        #Get the running status
        describe_response=client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])
    except Exception as e:
        print(e)

    print('Done...')

def main():
    project_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-identifica-libros/1619278513678'
    model_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-identifica-libros/version/demo-identifica-libros.2021-04-24T17.44.06/1619279040372'
    min_inference_units=1
    version_name='demo-identifica-libros.2021-04-24T17.44.06'
    start_model(project_arn, model_arn, version_name, min_inference_units)

if __name__ == "__main__":
    main()
```

Figura A-9: Código del programa istart.py.

A.10. Istop.py.

Este programa es proporcionado por Amazon Rekognition, su ejecución permite detener el modelo de aprendizaje. Hace uso de los módulos *boto3* y *time* de Python, se detiene el modelo con el ARN que la función *stop_model* recibe como parámetro, el programa cambia el estado del modelo cambie a deteniendo, al terminar notifica mediante Shell que se ha realizado correctamente.

```python
import boto3
import time


def stop_model(model_arn):

    session=boto3.Session(profile_name='ricardo')
    client=session.client('rekognition')

    print('Stopping model:' + model_arn)

    #Stop the model
    try:
        response=client.stop_project_version(ProjectVersionArn=model_arn)
        status=response['Status']
        print ('Status: ' + status)
    except Exception as e:
        print(e)

    print('Done...')

def main():

    model_arn='arn:aws:rekognition:us-east-2:752892782220:project/demo-identifica-libros/version/demo-identifica-libros.2021-04-24T17.44.06/1619279040372'
    stop_model(model_arn)

if __name__ == "__main__":
    main()
```

Figura A-10: Código del programa istop.py.

A.11. Icustom.py.

El programa **icustom.py**, importa el módulo *boto3* de Python. Se copia el ARN del modelo, el bucket donde se encuentra el dataset de imágenes y la foto que se desea analizar.

Mediante la función *client* del módulo *boto3*, permite indicar que se desea usar el servicio Amazon Rekognition ***('rekognition')***, haciendo referencia a que se utilizará

dicha API. La función *detect_custom_labels* pasa como parámetro el ARN del modelo y la imagen en bytes del bucket S3. Para finalizar mediante un bucle *for*, se muestran todas las etiquetas detectadas por Rekognition.

```
import image
import boto3
#from pprint import pprint
#Custom labels
model='arn:aws:rekognition:us-east-2:752892782220:project/demo-identifica-libros/version
/demo-identifica-libros.2021-04-24T17.44.06/1619279040372'
bucket='demo-libros-llaves'
photo='Libros a identificar/Para identificar/IMG_5002.JPG'

session = boto3.Session(profile_name='ricardo')
client = session.client('rekognition')
rekresp = client.detect_custom_labels(ProjectVersionArn=model, Image={'S3Object':{'Bucke
t':bucket,'Name':photo}})
#pprint(rekresp)
print("El libro que se ha identificado en la imagen es:")
for label in rekresp['CustomLabels']:
    print(label['Name'],", con un porcentaje del: ",label['Confidence'],"%")
```

Figura A-11: Código del programa icustom.py.

A.12. Versión 2 icustom.py.

El programa **icustom.py** se ha modificado para poder tomar fotos mediante el uso de una cámara integrada a la Raspberry pi, para ello se ha importado el módulo *picamera* y *time* de Python. La función PiCamera() permite identificar la camera para poder iniciarla mediante la función *start_preview()*, se utiliza la función *sleep(10)* de time para que el sistema espera 10 segundos antes de tomar la foto, la función *capture()* guarda la imagen en la dirección que se indique como parámetro y se finaliza el ciclo de la cámara con *stop_preview()*.

Para que Amazon Rekognition pueda enviar la imagen al algoritmo para que este realice el reconocimiento del objeto, es necesario que la foto se encuentre en el bucket de Amazon S3, esto se realiza creando un nuevo cliente con el módulo *boto3*, pero indicándole que se desea usar el servicio S3 pasándole el parámetro a la función ***client ('s3').***

La función open de Python nos permite abrir la captura que se ha realizado anteriormente y guarda la imagen en bytes en la variable *data*. El cliente s3 permite cargar la imagen al bucket mediante la función *upload_fileobj()*, pasándole como parámetro los datos de la imagen, el bucket y la ruta, con el nombre que se desea guardar la imagen. El resto del programa coincide con el que se ha creado en el Anexo **A.11**.

```
import image
import boto3
import picamera
from time import sleep
#Custom labels

model='arn:aws:rekognition:us-east-2:752892782220:project/demo-identifica-libros/version/demo-ide
ntifica-libros.2021-04-24T17.44.06/1619279040372'
bucket='demo-libros-llaves'
photo='Raspberry/IMAGEN.JPG'

#Camera
camera=picamera.PiCamera()
camera.start_preview()
sleep(10)
camera.capture('/home/pi/Desktop/image.jpg')
s3=boto3.client('s3')
with open('/home/pi/Desktop/image.jpg','rb') as data:
    s3.upload_fileobj(data,bucket,photo)
camera.stop_preview()

#Rekognition
client = boto3.client('rekognition')
rekresp = client.detect_custom_labels(ProjectVersionArn=model, Image={'S3Object':{'Bucket':bucket
,'Name':photo}})
print("El libro que se ha identificado en la imagen es:")
for label in rekresp['CustomLabels']:
    print(label['Name'],", con un porcentaje del: ",label['Confidence'],"%")
```

Figura A-12: Código del programa icustom.py.

B. Anexo II – Casos de uso Amazon Rekognition.

En este anexo, se mencionarán diversos casos de uso reales que permitirán comprender qué objetivos se pueden alcanzar con Amazon Rekognition.

B.1. Clasificación de Legos.

Mediante el uso de etiquetas personalizadas de Amazon Rekognition, Mike Chambers ha logrado detectar 250 ladrillos de LEGO diferentes. Ha entrenado con 4374 imágenes el modelo y mediante una aplicación móvil que ha creado, al tomar una foto del Lego reconoce el objeto he indica el porcentaje.

Disponible en: https://www.youtube.com/watch?v=5uNs7JFY9Tk.

B.2. Rayos X.

Con las etiquetas personalizadas se ha creado un modelo que permita detectar información a partir de datos de rayos X que permita identificar anomalías rápidamente y con una inversión de recursos y de bajo costo.

Disponible en: https://towardsdatascience.com/augment-pneumonia-diagnosis-with-amazon-rekognition-custom-labels-6f2fb79a986d.

B.3. Clasificador de flores naturales.

Se crea un clasificador de flores naturales que haciendo uso de las etiquetas personalizadas y haciendo uso del conjunto de datos Oxford Flower 102, donde con más de 8189 imágenes se ha podido crear un modelo capaz de clasificar las flores en las 102 categorías que tiene.

Disponible en: https://pub.towardsai.net/build-natural-flower-classifier-using-amazon-rekognition-custom-labels-8aabc3c8931c.

B.4. Inventario de refrigeradores.

Haciendo uso de Amazon Rekognition y AWS DeepLens, Chris Miller y Siaterlis Konstantinos han sido capaces de construir un verificador de inventario de refrigeradores, lo que permite que la Inteligencia Artificial haga el trabajo de realizar la lista de la compra.

Disponible en: https://dev.to/aws-builders/what-s-in-my-fridge-36o2.

B.5. Ropa en el tendedero con Arduino.

Haciendo uso de una cámara Arduino, AWS Lambda y Rekognition y Amazon Echo, que permita verificar si se ha dejado la ropa en el tendedero. El usuario consulta a Alexa si ha dejado la ropa en el tendedero, con la función lambda de Alexa se solicita una imagen al Arduino, recibida la imagen la envía a Rekognition y compara con las etiquetas devueltas para determinar si se encontró algún objeto en la imagen.

Disponible en: https://www.hackster.io/xelfer/bring-in-the-washing-b40fba

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá