Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Ingeniería en Sistemas Computacionales

Desarrollo de software para el reconocimiento de edificios utilizando Deep Learning

Harry Daniel Vargas Pérez

1150533

Asesor: Ing. Ignacio Cruz Domínguez

Montemorelos, Nuevo León, México

22 de abril de 2021

Resumen

Múltiples empresas y organizaciones ofrecen visitas guiadas sobre sus instalaciones para que los visitantes puedan interesarse en conocer más y llevarse una buena impresión. Para llevar a cabo esta tarea, se requiere de recursos económicos y tiempo en capacitar a recurso humano para hacer una actividad que se vuelve monótona. El objetivo de esta investigación es contar con una herramienta que permita proporcionar ayuda al área de promoción y reclutamiento en proporcionar información a los visitantes sobre los edificios que ellos observan, además de implementar inteligencia artificial para crear un reconocedor de edificios, mediante imágenes que ellos capturen. Dicha aplicación busca ser funcional tanto en iOS y Android, para tener una mayor compatibilidad y alcance, el framework de desarrollo de esta aplicación es React Native que utiliza JavaScript.

Palabras Clave

Dataset, API, Framework, Inteligencia Artificial

ÍNDICE

Desarrollo de software para el reconocimiento de edificios utilizando Deep Learning

Harry Daniel Vargas Pérez e Ignacio Cruz Domínguez

Facultad de Ingeniería y Tecnología, Universidad de Montemorelos, México

I. INTRODUCCIÓN

A. Antecedentes

Diariamente las empresas reciben multitud de visitas, algunas de ellas de manera personal, otras de tipo empresarial y otras más de tipo comercial, ya sea en mayor o menor medida. Independientemente de su formalidad y relevancia, la empresa puede aprovechar estas visitas para generar cercanía y potenciar su marca con posibles clientes a futuro dependiendo del trato y la impresión que causen. En beneficio de la imagen de la empresa, las visitas deben ser atendidas de acuerdo a las características propias de la empresa u organización para dar una impresión correcta del funcionamiento de la misma.

Para atender a las visitas, las organizaciones dedican distintos recursos como tiempo, dinero y esfuerzo, además de requerir una preparación del recurso humano para poder brindar una atención adecuada y de calidad según sea la visita que reciban. A los visitantes les interesa la información acerca de lo que observan durante el recorrido, como los principales edificios de la organización o los espacios en los que se encuentran.

Distintas universidades también ofrecen recorridos por sus instalaciones para que posibles nuevos alumnos se interesen en realizar su preparación académica con ellos; además de otros recorridos a grupos en espacios como bibliotecas o facultades específicas que dan más ayuda al interesado o interesados a poder hacer una evaluación más completa e integral de la universidad que están visitando. También algunas universidades están optando por ofrecer un recorrido virtual a través de sus instalaciones [1], [2], [3], esta es una gran oportunidad que puede ser aprovechada más en el tiempo que se está viviendo debido al confinamiento por causa del COVID-19; este recorrido atraviesa los lugares más referentes del plantel mediante un video o un mapa utilizando *Google Maps* para poder desplazarse sobre las áreas determinadas. Existen plataformas como civitatis [4] que ofrecen recorridos a lugares alrededor del mundo de manera presencial que incluyen visitas a universidades, museos, zonas turísticas, entre otras. Dicha plataforma también está bajo las recomendaciones e instrucciones de parte del personal médico en cada lugar.

B. Definición del problema

Con la Revolución Industrial 4.0 [5], la necesidad de implementar tecnología en las organzaciones ha impulsado también al Internet de las Cosas (IoT por sus siglas en inglés *Internet of Things*), donde básicamente todo con lo que nos relacionamos estaría conectado a internet, almacenando datos a la nube [6]. Actualmente muchos de los procesos realizados en la universidad pudiesen ser optimizados mediante la implementación de tecnología, uno de los que se observaron para la realización de esta investigación es el acompañamiento en los recorridos guiados a través del campus. Al realizar un recorrido es interesante e importante para el visitante, obtener información y realimentación acerca de los principales lugares que se visitan. En este proyecto se desarrolla una aplicación móvil que provee esta información complementaria acerca de los principales edificios e instalaciones de la Universidad de Montemorelos. Se implementa también el reconocimiento de edificios mediante imágenes capturadas por el usuario, en esta investigación se trabajó con 8 edificios, recolectando imágenes de sus fachadas en distintos ángulos y condiciones climáticas.

C. Justificación

Aprovechando el auge y accesibilidad de los dispositivos móviles y basado en la necesidad presentada, en este trabajo se desarrolla un software que ayuda en la guía de recorridos, al identificar principales sitios, edificios e instalaciones, tarea que es realizada por personal humano de manera continua. Esta actividad se ve pausada momentáneamente debido al confinamiento impuesto por las autoridades sanitaria por causa de la pandemia del COVID-19. Se desarrolla una aplicación móvil que funcione en las plataformas de Android y iOS, que permite identificar los departamentos y facultades dentro de la institución utilizando algoritmos de Inteligencia Artificial y Deep Learning para conseguir una predicción aceptable.

En esta aplicación el usuario hace uso de la cámara para poder capturar una imagen del edificio sobre el cual se requiere información y tomarla como entrada. Para proveer ésta realimentación se utilizan técnicas de reconocimiento basadas en aprendizaje profundo *(Deep Learning)* para lograr una identificación aceptable. La implementación de esta característica permite lograr una cercanía a los valores e información importante de la institución.

La aplicación es relevante porque permite que los nuevos visitantes o grupo de visitantes, puedan tener información adicional además de la provista por el personal que guía el recorrido, el cual tiene que ser instruido y capacitado previamente. Este proceso requiere invertir tiempo de preparación en conocer bien los lugares e historias. En ocasiones, para el personal guía, puede resultar monótono repetir las descripciones e información, lo cual no sucederá con la aplicación.

Harry Vargas es estudiante de la Facultad de Ingeniería y Tecnología en la Universidad de Montemorelos, e-mail: 1150533@alumno.um.edu.mx.

Ignacio Cruz, Ing. es profesor en la Facultad de Ingeniería y Tecnología en la Universidad de Montemorelos., Nuevo Léon, Mexico, e-mail: icruz@um.edu.mx.

Ésta aplicación es útil para el departamento de promoción y reclutamiento en la Universidad de Montemorelos al llevar a cabo su actividad de guía que es realizada de forma recurrente, facilitando que los visitantes obtengan la información de los edificios a partir de una entrada visual mediante la cámara del dispositivo móvil. La información provista por la aplicación móvil enriquece la experiencia y la relación con la institución. El visitante puede obtener información relevante, no sensible o pública de cada departamento o facultad como horario de atención, teléfono de contacto, las carreras con las que cuenta cada facultad, historia, entre otras.

D. Objetivos

1) Objetivos generales: Se espera obtener una aplicación que permita ofrecer ayuda a personas interesadas en conocer la Universidad de Montemorelos, sumado a esto la disminución del trabajo de acompañamiento en los recorridos y la oportunidad de hacer más didáctico y tecnológico el recorrido a la institución. Esta innovación permitiría a la universidad contar con una aplicación descriptiva de sus instalaciones, además de la oportunidad de compartir la información específica de los horarios de atención en cada una e información de contacto.

2) Objetivos específicos:

- Identificar instalaciones y edificios importantes del campus de la Universidad de Montemorelos mediante técnicas de Deep Learning, a partir de la entrada proveniente de la cámara del dispositivo móvil del usuario.
- Construir un clasificador que toma la imagen y predice a qué facultad o edificio corresponde.
- Describir el proceso de clasificación e identificación, además de proveer medidas del rendimiento y pruebas.
- Desarrollar una aplicación que utiliza el modelo obtenido utilizando React Native.
- Describir las herramientas de desarrollo y el proceso llevado a cabo en la integración

II. FUNDAMENTOS TEÓRICOS

A. MARCO TEÓRICO

En ésta sección son presentados algunos conceptos fundamentales en esta investigación, además de herramientas utilizadas en el desarrollo del mismo. En su primera parte se muestran conceptos relacionados a el modelo creado capaz de reconocer imágenes y predecir a qué edificio se está apuntando en la fotografía, además de algunas herramientas utilizadas para su desarrollo e implementación. En la segunda parte se explican conceptos y herramientas para la creación del servicio capaz de almacenar la información que se le devolverá al usuario y por último las herramientas que fueron usadas para la unificación de las partes anteriores en una aplicación móvil.

1) Inteligencia Artificial:

La Inteligencia Artificial (IA) es parte de las ciencias de la computación que se encarga de los diseños de sistemas inteligentes, los cuales contienen características que son relacionadas con la inteligencia en la conducta humana [7]. La IA ha tenido múltiples aplicaciones no solo en la informática o robótica, también está presente actualmente en su implementación en áreas empresariales para poder realizar predicciones de resultados en tiempo real y también para procesar gran cantidad de datos. A su vez, la inteligencia artificial ha tenido un gran progreso gracias al aumento en los recursos económicos, tecnológicos y de datos para poder trabajar y realizar más investigaciones en este campo[8].

2) Machine Learning:

El aprendizaje automático *(Machine Learning)* es una herramienta de la inteligencia artificial que se enfoca en el análisis de datos para poder obtener predicciones a futuro sobre posibles comportamientos. Para esto se utilizan algoritmos que utilizan los datos pasados o presentes y ayudan a formular una predicción. Hay tres tipos de aprendizaje automático:

- Aprendizaje supervisado: Se trabaja con datos etiquetados o clasificados.
- Aprendizaje no supervisado: Se utilizan datos en bruto, sin etiquetas, lo que lleva a realizar agrupaciones buscando similitudes entre los datos.
- Aprendizaje semi-supervisado: No todos los datos a utilizar están etiquetados.
- Aprendizaje de refuerzo: es cuando los datos no están etiquetados, pero pasado un periodo y unas acciones, el algortimo será retroalimentado mediante actualizaciones.

Esta rama de la inteligencia artificial tiene un gran potencial y es utilizada en distintas áreas hoy en día como la medicina, finanzas, educación, entre otras [9].

3) Deep Learning:

El *deep learning* (aprendizaje profundo) se asocia a la idea de "imitar el celebro a partir del uso de hardware y software, para crear una inteligencia artificial pura", la cual usa una capacidad de abstracción jerárquica para poder separar los datos de entrada y poder obtener características de los datos de un nivel de complejidad alto [10]. En otra definición más sencilla de entender, el deep learning es el aprendizaje que una máquina puede hacer para cumplir con un determinado objetivo, obteniendo datos ocultos o profundos desde una red neuronal [11].

4) Redes Neuronales:

Una de las definiciones que se puede utilizar para dar significado a las redes neuronales artificiales puede ser: "Una red neuronal es un modelo computacional, paralelo, compuesto de unidades procesadoras adaptativas con una alta interconexión entre ellas" [12] Básicamente la comparativa con una red neuronal biológica se refleja en la distribución de las operaciones que se realizan en el conjunto de los elementos básicos que representan las neuronas. Estas neuronas están interconectadas entre sí y se denominan pesos sinápticos, estos pesos varían con el paso del tiempo de entrenamiento. Así se puede definir el aprendizaje de una red neuronal como el proceso en el cual se modifican las conexiones entre las neuronas para poder realizar un objetivo definido.

Las redes neuronales artificiales tienen una topología que consiste en la forma en que las neuronas están organizadas, de manera que crean agrupaciones (capas) más cerca de la entrada o de la salida de la red. Algunos de los parámetros fundamentales en estas redes son el número de capas formadas, la cantidad de neuronas en cada capa, el grado de conectividad y el tipo de conexiones entre las neuronas [13].

5) Python:

Python [14] es un lenguaje dinámico y fuertemente tipado, además es orientado a objetos [15]. Python presenta ventajas ante otros lenguajes de programación como:

- Es un lenguaje expresivo, el código suele ser más corto que utilizando otros lenguajes, además de ser considerado un lenguaje de alto nivel.
- La lectura del código resulta fácilmente legible, ya que su sintaxis permite escribir de manera elegante.
- Python posee múltiples estructuras de datos que pueden ser utilizadas de manera simple.

Estas son algunas de las ventajas que presenta trabajar con Python, además de su continua actualización debido al desarrollo de mantenimiento que tiene [16].

6) Anaconda:

Anaconda [17] es una interfaz gráfica de usuario *(GUI)* que permite iniciar aplicaciones y administrar fácilmente paquetes, entornos y canales de conda sin usar la línea de comandos [18]. Está disponible tanto para Windows, como para MacOS y Linux. Anaconda cuenta con distintos paquetes, en esta investigación se utilizó la versión *"Individual Edition"* [19]. Este paquete cuenta con *Anaconda Navigator* la cual es una *GUI* de escritorio, que permite la creación y administración de ambientes, además de la instalación de algunas herramientas de desarrollo como lo es *Jupyter Notebook* [20], que es una aplicación web que permite la escritura de código en Python y es instalada predeterminadamente junto con Anaconda. Dicha herramienta es la utilizada para el desarrollo de la parte de inteligencia artificial en este proyecto.

7) Tensorflow:

Tensorflow [21] es una librería de software de código abierto que es usada para implementar sistemas de aprendizaje automático [15]. Esta plataforma fue desarrollada por Google para compartir con otros investigadores y poder desarrollar sistemas capaces de crear y entrenar redes neuronales para reconocer patrones; además de que provee distintas herramientas, librerías y recursos de la comunidad para poder realizar innovaciones en el campo del aprendizaje automático. Tensorflow es capaz de ser utilizado en distintos lenguajes de programación, en esta investigación es utilizado con el lenguaje Python y es instalado dentro de un ambiente gestionado por Anaconda.

8) Métricas de Evaluación:

Mediante el uso de la herramienta de Tensorflow, permite la creación de proyectos de *machine learning*. Se necesita una métrica de evaluación adecuada para evaluar el desempeño de un modelo en la tarea de reconocimiento de imágenes, se utiliza el término *accuracy* para referirse a esta exactitud con la que el modelo predice los resultados. Al compilar el modelo obtenido se asigna esta métrica, mientras que también se utilizan otros dos componentes para configurar el proceso de aprendizaje de una red neuronal que son la función de pérdida *(loss function)* y el optimizador, estos dos componentes deben de elegirse de acuerdo a la tarea que se realice y tomar en cuenta otros factores como el número de clases o los posibles desequilibrios de clases [22].

9) JavaScript:

JavaScript *(JS)* es un lenguaje de programación muy utilizado en los últimos años debido a su flexibilidad, es un lenguaje multiparadigma y muchos de los sitios web lo utilizan para las animaciones y más recientemente para back-end, trabajando del lado del servidor [23]. Es un lenguaje de script, lo que significa que no ocupa ser compilado, solamente interpretado [24].

10) API REST:

Se denomina API por sus siglas en inglés *Application Programming Interface*. Es un conjunto de métodos o *endpoints* que son ofrecidos por una biblioteca de código, que será utilizado por otro software, demostrando así la capacidad de comunicación entre distintos componentes de software. REST también es un acrónimo en inglés *(Representational State Transfer)* y refiere a un tipo de arquitectura que sigue cuatro principios fundamentales:

- Incluye el uso de métodos HTTP: aquí se establece una relación uno-a-uno entre los métodos HTTP y las operaciones de un CRUD *(Create, Read, Update y Delete)*, siendo el verbo POST utilizado para crear un registro, GET para obtenerlo, PUT para actualizar y DELETE para eliminarlo. Estas 4 funciones básicas son importantes ya que engloban las acciones necesarias al manejar información.
- No mantiene estados (stateless): cada petición debe de ser completa e independiente, que contenga todos los datos para ser completada la orden.
- Utiliza URIs en forma de directorio para realizar una operación definida: las rutas definidas para los endpoints deben de ser intuitivas, se utiliza en estilo de directorio para reconocer la jerarquía.
- Transfiere XML, JSON o ambos: el formato JSON *(JavaScript Object Notation)* es un formato de archivo simple para describir datos jerárquicamente [25], [26].

11) Node.js:

En su sitio web oficial, se define a Node.js como un entorno de ejecución de JavaScript E/S asíncrono y que está diseñado para crear aplicaciones de red escalables [27]. Utiliza el motor de JavaScript V8 de Google [26]. Entre sus características destaca que es de código abierto y multiplataforma, es decir, que puede ser utilizado por distintos sistemas como Windows, Linux, MacOS, entre otros. El lenguaje utilizado en Node.js es JavaScript puro, con la ventaja que permite usar dicho lenguaje del lado del cliente y también del servidor [23]. Node.js es utilizado para el desarrollo del *back-end* en diversos proyectos actualmente.

Express.js [28] es un marco web utilizado en el desarrollo de aplicaciones web minimalistas y comúnmente utilizado para la creación de APIs *(Application Programming Interface)*, ya que crea un servidor de HTTP que permite habilitar un puerto para la comunicación.

Mongoose [29] es una herramienta para el modelado de objetos de MongoDB, soporta promesas y llamadas de respuesta *(callbacks)*. También funciona para crear una conexión con la base de datos.

12) Base de Datos noSQL:

Las bases de datos NoSQL, son bases de datos de código abierto, sin esquema, escalables horizontalmente (se pueden agregar más servidores para trabajar en la misma carga de trabajo) y de alto rendimiento. Estas características las hacen muy diferentes de las bases de datos relacionales, la opción tradicional para datos espaciales. Las bases de datos NoSQL están bien adaptadas para manejar los desafíos típicos de big data, incluidos el volumen, la variedad y la velocidad. Por estas razones, las industrias privadas los adoptan cada vez más y se utilizan en la investigación. Se ha hecho más común utilizar las base de datos NoSQL en los últimos años debido a su capacidad para administrar datos no estructurados (por ejemplo, datos de redes sociales) [30].

13) MongoDB Atlas:

Esta herramienta ofrece una base de datos para aplicaciones modernas como un servicio en la nube totalmente automatizado diseñado y ejecutado por el mismo equipo que crea la base de datos. Utilizando esta herramienta (panel de control y la API fáciles de usar), le permiten al desarrollador dedicar más tiempo a crear aplicaciones y menos a administrar su base de datos. La utilización de MongoDB Atlas presenta una gran ayuda al poder desplegar la base de datos y ser accedida desde cualquier lugar, gracias a que se aloja en sus servidores y provee también configuraciones para seguridad [31], [32].

14) Flask :

Flask [33] es un framework pequeño y ligero, lo suficiente para ser denominado microframework, para aplicaciones web construido sobre Python, que proporciona un framework eficiente para crear aplicaciones basadas en web utilizando la flexibilidad de Python y un fuerte apoyo de la comunidad con la capacidad de escalar para servir a millones de usuarios [34]. Flask cuenta con un excelente soporte comunitario, documentación y bibliotecas de apoyo. Para desplegar una API en un servicio como Heroku, es necesario contar con un archivo de configuración que indicará el comando que se ejecutará para poder iniciar la aplicación, este archivo se denomina *Procfile* y se utiliza Gunicorn [35]para poder crear el servidor y ejecutar el script [36].

15) React:

En su sitio web [37], se define a React como una biblioteca de JavaScript para construir interfaces de usuario interactivas de forma sencilla. Se utiliza para manejar la capa de visualización en aplicaciones de una sola página y desarrollo de aplicaciones móviles. Es mantenido por Facebook, Instagram y una comunidad de desarrolladores y corporaciones. React se esfuerza por proporcionar velocidad, simplicidad y escalabilidad. Algunas de sus características más notables son JSX, componentes con estado, modelo de objeto de documento virtual [38].

16) React Native :

React Native [39] es un framework desarrollado por Facebook, originalmente fue publicado para desarrollar en iOS, después se introdujo el soporte para también poder desarrollar en Android. El propósito principal de React Native es permitirle al programador hacer un solo desarrollo que funcione en ambas plataformas, se ejecuta en JavaScriptCore (iOS) o V8 (Android) dentro de las aplicaciones y se procesa en componentes específicos de la plataforma de nivel superior [40].

Expo es una cadena de herramientas construida alrededor de React Native para ayudar a los desarrolladores a comenzar sus proyectos rápidamente. Expo proporciona un conjunto de herramientas y servicios para desarrollar, construir, implementar, probar o incluso ejecutar simuladores en la plataforma específica desde la misma base de código. Específicamente, ofrece una colección de soluciones listas como acelerómetro del dispositivo, cámara, notificaciones, geolocalización, etc. Un punto conveniente al usar Expo es que los desarrolladores no necesitan descargar XCode o Android Studio para ejecutar la aplicación. Pueden ejecutar la aplicación directamente con Expo CLI [41]s.

17) Heroku :

Heroku permite implementar, ejecutar y administrar aplicaciones escritas en Ruby, Node.js, Java, Python, Clojure, Scala, Go y PHP. Una aplicación es una colección de código fuente escrito en uno de estos lenguajes, y alguna descripción de dependencia que instruye a un sistema de compilación sobre qué dependencias adicionales se necesitan para compilar y ejecutar la aplicación. Heroku es una plataforma políglota: le permite crear, ejecutar y escalar aplicaciones de manera similar en todos los idiomas, utilizando las dependencias y Procfile. Procfile expone un aspecto arquitectónico de la aplicación y esta arquitectura le permite escalar cada parte de forma

independiente. Heroku ejecuta la aplicación dentro de un dyno (contenedor ligero de Linux) [42].

B. ESTADO DEL ARTE

A continuación, se presentan algunas investigaciones que se encontraron relacionadas a este proyecto de investigación.

1) CIU TOUR:

En una investigación realizada en la Universidad de Valencia, España [43], se llevó a cabo un proyecto similar a este, donde crearon una aplicación móvil capaz de tomar una foto del entorno (ya sea un edificio o un monumento) en su dispositivo móvil y utilizar un modelo de inteligencia artificial creado con las herramientas provistas por Tensorflow para poder clasificar la imagen si correspondía a un edificio o a un monumento al interior de su campus universitario, además de devolver más información relacionada que pudiera ser de relevancia para el usuario, esto con el fin de poder ayudar a un turista a completer un recorrido por su cuenta en las instalaciones.

En este trabajo se dio un enfoque a turistas que deseen hacer un tour de manera propia y con una buena precisión acerca de la información visualizada. En su aplicación incluyeron también un mapa de su universidad utilizando Google Maps, además de un menú donde se podía encontrar la información sobre los edificios o monumentos sin necesidad de hacer un reconocimiento mediante una fotografía. También cabe destacar que se realizó pensado en el sistema operativo de Android.

2) Building Recognition:

Otra investigación similar es una realizada en Perú [44], donde recolectaron un total de 2000 imágenes sobre 14 principales monumentos históricos y proponer una topología de red neuronal capaz de lograr un buen rendimiento al clasificar dichas imágenes, entendiendo que la tarea de clasificar este tipo de imágenes presenta un gran reto porque las fotografías pueden ser capturadas desde ángulos distintos, en condiciones climáticas distintas, entre otras cosas.

Su topología presentó buen rendimiento frente a otras que fueron evaluadas, sin embargo, destacan algunos puntos en donde se pueden enfocar en otros trabajos a futuro como lo son la implementación de más clases y confirmar que su modelo presenta buen rendimiento frente a otros tipos de modelos.

III. DESARROLLO

Para el desarrollo de este proyecto se dividieron en etapas para tener entregables del desarrollo de manera controlada, los tres primeros puntos se explica el desarrollo realizado para la creación del modelo de inteligencia artificial. Posteriormente, los siguientes dos puntos tratan sobre la solución propuesta para el manejo de la información adicional que será entregada al usuario junto con la predicción del modelo. Y por último, la implementación de todo esto en una sola aplicación móvil

A continuación, se muestran los siguientes puntos para el desarrollo del proyecto:

A. Creación del dataset

Para poder crear un modelo de inteligencia artificial capaz de reconocer los edificios en la Universidad de Montemorelos, es necesario previamente recolectar imágenes que el modelo necesitará para poder entrenarse y ser capaz de reconocer edificios.

1) Recolección de imágenes: Las imágenes utilizadas son de la fachada de los edificios como las facultades, departamentos y otros espacios importantes, esto porque es lo que el usuario verá cuando desee saber una predicción del lugar. Se hizo la recolección de 50 imágenes de cada edificio (clase), siendo 8 edificios reconocidos. Esto hace un total de 400 imágenes capturadas para crear el modelo.

Estas imágenes son fotografías tomadas por un dispositivo móvil (iPhone 12 Pro Max), esto debido a que el modelo sería utilizado en una aplicación móvil y las imágenes con las que sería probado el modelo serían capturadas por dispositivos móviles. Todas las imágenes fueron tomadas en distintos ángulos, altura y condiciones del clima en dirección a la fachada del edificio; todas las imágenes fueron tomadas durante el día.

2) Organización de las imágenes: Las imágenes fueron exportadas a formato .jpeg para poder ser utilizadas en el modelo realizado. Todas las imágenes fueron separadas en distintos directorios (carpetas), el nombre de la carpeta es dependiendo del edificio al que pertenecen las fotografías.

B. Creación de modelo de IA

Para poder reconocer edificos en una imagen, se crea un modelo de clasificación con redes neuronales convolucionales profundas. A continuación se describe la manera en como fue creado y entrenado el modelo:

1) Preparación del entorno Python: Para la realización de esta investigación se creó un ambiente de desarrollo en Anaconda 1.10.0. Junto con esto, se instala la versión 3.7.10 del lenguaje de programación Python que es utilizada en el desarrollo del código fuente, también la versión 2.3.0 de tensorflow que permite la creación de la red neuronal, además de instalar TensorflowJS en su versión 3.3.0 para exportar el modelo y ser utilizado en la aplicación. Para la escritura del código se utilizó la herramienta Jupyter Notebook contenida por Anaconda. Dentro de Tensorflow se encuentra la librería de Keras que ha sido utilizada para hacer uso de las capas utilizadas en el modelo.

El proyecto está estructurado en un directorio, en él se encuentra un archivo .ipynb (formato de Jupyter Notebook) con el código que recolecta el dataset, diseña la red neuronal y entrena el modelo que posteriormente es exportado. En una

carpeta al interior del directorio principal se encuentran las imágenes a utilizar para el entrenamiento del modelo, descrito previamente en el apartado 3.1.

2) Carga del dataset: Para cargar las imágenes contenidas en un folder dentro del directorio, se utiliza una herramienta contenida dentro del paquete de keras, esta herramienta permite particionar el total de imágenes recolectadas entre un paquete de entrenamiento y uno de prueba, el procentaje de división es 80% para entrenamiento y 20% para pruebas. Dichas imágenes son tomadas en una dimensión fija (224 x 224).

3) Estructura de la red neuronal: Para crear la red neuronal se utiliza la estructura Secuencial contenida dentro de Keras. Se comienza con una capa donde se aplica una rotación, un zoom y una vuelta aleatoriamente; posteriormente la siguiente capa cambia de tamaño a la imagen para ajustarlo a las dimensiones utilizadas en el experimento (224 x 224 píxeles).

A continuación, se utiliza una capa convolucional con 16 neuronas, esto permite crear un núcleo de convolución con la salida de la capa anterior como entrada y obtener un tensor de salidas, en otras palabras, la convolución consiste en ir comparando la imagen de entrada con una matriz creada (kernel). Dentro de esta capa se utiliza la función de activación ReLU *(rectified linear activation unit)* la cual, cuando recibe una entrada negativa, devuelve 0 y en caso contrario, el mismo valor positivo de la entrada, esto para la comparación con el kernel. Pasada la convolución se implementa una capa de *Max Pooling 2D* para hacer una reducción de las dimensiones de alto y ancho de la salida de la capa convolucional anterior, pero no modifica su profundidad.

Posteriormente, se vuelve a crear otra capa convolucional seguida de una capa de *max pooling*, pero en esa ocasión con 32 neuronas en la capa convolucional. A continuación, se implementa una capa *Dropout* para desactivar neuronas aleatorias y evitar que estas neuronas creen dependencia entre sí. Después, se agrega una capa *Flatten* para coonvertir los parámetros que vienen en forma de matriz a un arreglo vector, para posteriormente crear una capa *Dense*, es decir, completamente conectada con una dimensionalidad de salida de 128 y utilizando también la función de activación ReLu. Al final se agrega otra capa Dense con una salida del número total de clases para obtener la predicción. Esta descripción puede ser observada en la Figura 1, donde se muestran las capas contenidas dentro del modelo.

Después de esto, el modelo es compilado utilizando la función de coste *"Sparse Categorical Crossentropy"* para poder tratar el valor entre el estimado y el real, esto para optimizar el resultado. Además, se usa el optimizador Adam para la función del error y el *accuracy* (exactitud) como métrica de evaluación.

4) Pruebas: Las pruebas realizadas implican mover los hiperparámetros con los que se realiza el experimento, estos son las dimensiones de la imagen, el tamaño del lote o el número de épocas de entrenamiento. La variación de estos valores presenta un movimiento en los resultados al momento de entrenar, los valores definidos y utilizados en el código son los que dieron un mejor desempeño al momento de entrenar y evaluar el modelo.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (None, 224, 224, 3)       0
_________________________________________________________________
sequential_1 (Sequential)    (None, 224, 224, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 224, 224, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 56, 56, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
dense (Dense)                (None, 128)               12845184
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 1032
=================================================================
Total params: 12,851,304
Trainable params: 12,851,304
Non-trainable params: 0
_________________________________________________________________
```

Figure 1. Modelo de la red neuronal

5) Entrenamiento del modelo: Para el entrenamiento del modelo creado se tomaron los dos grupos de imágenes creados, uno de entrenamiento y uno de prueba, además de indicar un total de 20 épocas de entrenamiento. El objetivo visible durante el entrenamiento debe ser que el *accuracy* incremente a medida que pasan las épocas y el error disminuya.

6) Exportación: Después de que el modelo es entrenado y que ofrezca buen desempeño en su evaluación, se exporta en dos archivos, el primero de ellos es en formato JSON el cual contiene la estructura del modelo y un archivo .h5 que contiene los pesos del modelo. Estos dos archivos serán utilizados en la sección siguiente.

C. Creación de API REST en FLASK

Para que la aplicación pudiese consumir el modelo de inteligencia artificial, se decidió crear una API que pudiese recibir la imágen y regresar la predicción del modelo. En esta sección se describen los pasos para su creación.

Para poder crear esta API, se utilizó el framework Flask en su versión 1.1.2, con lo cual permitiría tener una conexión con el modelo. En la figura 2 se puede apreciar la estructura del directorio que contiene esta API, es sencillo y consta de los dos archivos exportados del modelo explicados en la sección anterior. En el archivo *Procfile* se guarda la configuración donde se utiliza gunicorn para crear el servidor HTTP. Y por último el archivo API.py en el que se encuentra el endpoint que es consumido para la predicción, se recibe la imagen en base64 y se realiza la función del modelo que permite obtener un arreglo de las predicciones. De esta predicción numérica se obtiene el máximo valor y se compara con las etiquetas definidas previamente, se retorna en formato JSON

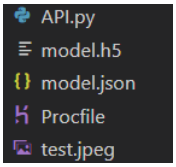

Figure 2. Estructura del API en Flask

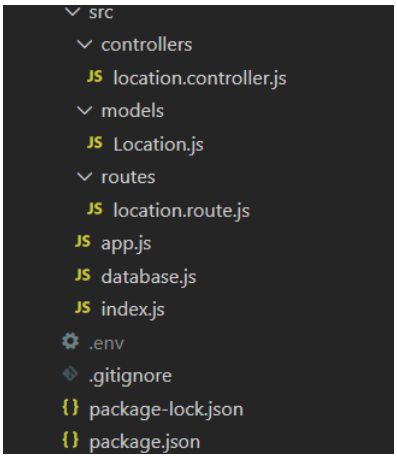

Figure 3. Estructura del API en Node.js

el nombre corto del edificio que se predijo para ser utilizada posteriormente en la aplicación.

D. Desarrollo de un API REST en Node.js

Uno de los objetivos del proyecto es poder devolver al usuario información adicional y específica de cada edificio. Para llevar a cabo esta tarea, se realizó una API Rest conectada a una base de datos que permitiría el manejo y visualización de la información complementaria.

1) Creación: Para la cración del API capaz de almacenar información y permitir a un administrador manejar dicha información, se utiliza Node.js en su versión 12.18.4 para poder crear dicha API que tendrá interacción con una base de datos alojada en MongoDB Atlas en su versión gratuita. Una vez finalizada, se utiliza Heroku para poder desplegar la API y poder acceder a ella desde cualquier lugar mediante un link generado.

En la figura 3, se muestra la manera en como está estructurado el código fuente desarrollado para el API. En la carpeta de *controllers* se encuentran los endpoints que serán utilizados para el manejo de la información. El archivo *Location.js* contiene el modelo con los campos requeridos para generar un nuevo registro de una ubicación. En el directorio de *routes* se encuentra el manejo de las rutas utilizando *express* para asignarle una dirección a cada endpoint. En el archivo *database.js* está la configuración a la base de datos creada previamente en la plataforma de MongoDB Atlas.

2) Función del servicio: Esta API será utilizada por la aplicación móvil y también por la plataforma web, siendo la app la que consumirá más la visualización de la información, mientras que la plataforma web será el medio por el cual pueda ser insertada nueva información o modificación de la misma.

Dentro de sus funciones, permite un *CRUD (Create, Read, Update y Delete)* completo para las ubicaciónes. Existen 3 endpoints sobre el obtener información, uno permite obtener todas las ubicaciones disponibles, otro permite una búsqueda específica por el id del registro creado y uno último que permite obtener información específica que será utilizada para cuando la app prediga sobre algún edificio reconocido.

También se cuenta la opción de crear, actualizar y eliminar un registro, estos últimos son utilizados en la plataforma web. Las respuestas regresadas son el formato JSON, esto permite un buen manejo de la información y también es sencillo de utilizar en los frameworks utilizados, ya que es común encontrarlo en las librerías disponibles más por ser de JavaScript, además de que en MongoDB se maneja en dicho formato igualmente.

E. Administración de ubicaciones

Se requirió de la creación de una plataforma para la administración de las ubicaciones e información contenida de las mismas, para esto se desarrolló una plataforma web utilizando el framework React en su versión 17.0.1 que a continuación será descrita.

1) Funcionalidad: Esta aplicación permite la visualización, modificación, eliminación y creación de registros de ubicaciones, mediante un formulario que proviene del modelo provisto por el API especificada en el punto anterior. Se utiliza axios para realizar las llamadas a los endpoints necesitados en el manejo de los datos.

El modelo de las ubicaciones incluye un nombre, un teléfono de contacto, sitio web en caso de contar con alguno, correo electrónico, el horario de atención, una imágen de la fachada y también las coordenadas. Toda esta información es la que se maneja y permite dar un plus al usuario que desconoce la universidad y quisiera contactarse con alguna facultad o departamento en específico.

En la pantalla principal mostrada en la Figura 3 se puede apreciar la manera en cómo se ve la lista de las ubicaciones ya registradas, las funcionalidades disponibles están marcadas por los botones de *"modificar"* y *"borrar"*, además de que al hacer click sobre cada tarjeta se puede ver más información sobre ese item. También se puede agregar un nuevo registro con el botón en la barra superior. Las imágenes no son guardadas en la base de datos, deben ser subidas previamente a una carpeta compartida en *Google Drive* y así poder generar un link que es el guardado en el registro del formulario.

Cabe mencionar que en esta versión la plataforma no cuenta con un login, por lo que no tiene seguridad al momento de administrar en los registros realizados.

F. Implementación del modelo en la app

Se utilizó el framework de react native que permite hacer un desarrollo móvil multiplataforma, en dicho framework se

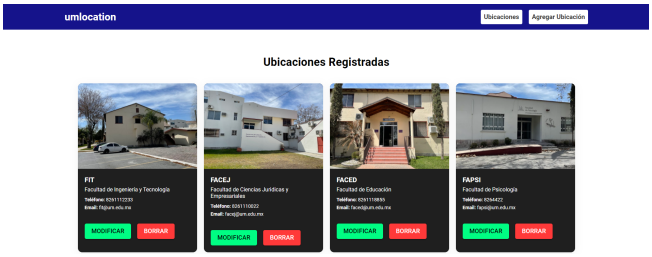

Figure 4. Interfaz principal plataforma web

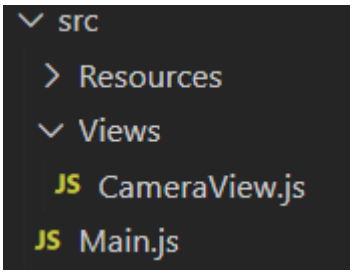

Figure 5. Estructura de la aplicación móvil

accede a los permisos de la cámara del dispositivo para poder tomar la fotografía del lugar y se hacen las llamadas a las dos API's.

En la figura 5 se muestra la estructura sencilla del código de la aplicación móvil utilizando react native. En el archivo CameraView.js se contiene todos los complementos necesarios para la aplicación, este archivo será llamado *archivo principal* en las siguientes subsecciones.

1) Configuración de la cámara: Dentro del archivo principal, se encuentran las validaciones para solicitar permisos al usuario para poder acceder a la cámara del dispositivo. Una vez que los permisos sean otrogados, en la vista será desplegada la imagen que está disponible para capturar. Se utiliza el componente de expo-camera que permite hacer las validaciones previas y la visualización de lo que capta la cámara trasera del dispositivo.

2) Predicción: Para obtener la predicción del modelo, se realiza una llamada al endpoint en el API que se realizó con Flask y se le envía la imagen capturada y convertida a base64. Esta llamada se realiza utilizando axios y se hace a una url desde donde está corriendo el API. En esta aproximación debido al tamaño del modelo se ejecutó de manera local y se compartió el puerto para poder acceder desde la aplicación. Una vez obtenida la respuesta del modelo, se guarda este resultado en un estado.

3) Despliegue de la información: Después de haber obtenido la respuesta de la predicción realizada por el modelo, se ejecuta otra llamada ahora al API de Node.js que contiene la información detallada del edificio. Dicha llamada se ejecuta también con axios al url donde está desplegada mediante Heroku, donde se accede a un endpoint creado específicamente para devolver solo cierta información y enviando como parámetro la predicción. Una vez realizado este proceso, se despliega en la pantalla la predicción de qué edificio reconoció el modelo y su información adicional.

Métrica	Entrenamiento	Validación
Accuracy	0.92	0.90
Loss Function	0.21	0.31

Table I
EVALUACIÓN DEL MODELO TRAS 20 ÉPOCAS DE ENTRENAMIENTO

Figure 6. Gráficas de entrenamiento del modelo

IV. RESULTADOS

El modelo realizado utilizando Keras obtuvo buenos resultados en su entrenamiento y con sus datos de validación, estos datos son un total de 320 imágenes para entrenar y 80 para validación. En la Tabla 1 se puede observar los resultados que se obtuvieron al final del entrenamiento y en la Figura 6 se muestra la gráfica de los resultados con el paso de las 20 épocas, se puede observar el incremento en el *accuracy* a medida que pasaban las épocas, mientras que el *loss* baja.

En la Figura 7 se puede observar una captura de cómo es que se ve implementada la aplicación en un dispositivo móvil, en este caso se hicieron las pruebas en iOS. En la parte superior de la pantalla se puede observar la información que se le es ofrecida al usuario que pudiese ser de interés, en la parte inferior se encuentra un botón de cámara que permite capturar una foto y hacer una predicción, arriba del botón de la cámara se encuentra un espacio blanco donde se muestra el titulo de la predicción del modelo. Se realizaron otras pruebas aparte ya con la aplicación funcionando en el dispositivo, donde clasificó de manera correcta 7 de 10 imágenes con las que fue puesto a prueba.

El tiempo de respuesta toma entre 3 a 4 segundos, esto debido a que hace dos llamadas a distintas API's. La aproximación de cargar el modelo en una API diferente fue a causa del incremento en el tamaño de la aplicación, debido a que el modelo ocupaba mucho espacio y los objetivos era conseguir tener una aplicación que fuera posible de usar en muchos dispositivos y no requiriera tantos atributos de

Figure 7. Interfaz principal de la aplicación móvil

hardware disponibles. La aproximación utilizada tuvo un buen desempeño y permitió enfrentar algunos aspectos de integración.

La plataforma desarrollada en React, permite a un administrador realizar las tareas de creación y registro de nuevas ubicaciones, así como la visualización de los datos guardados. La aplicación móvil muestra buen funcionamiento tanto en Android como en iOS, es simple pero cumple con el objetivo buscado donde se pueda predecir un resultado y devolver datos relevantes. Esta aplicación se ejecuta mediante expo, que permite correr esta app sin tener que instalarla previamente con un instalador, permitiendo así ser más sencillo su ejecución en modo de prueba.

V. CONCLUSIONES

Los productos presentados ofrecen una gran ayuda al cumplimiento de los objetivos, debido a que la implementación de todos sus componentes hacen a la aplicación completa. Se encontró que el modelo presentado ofrece buen rendimiento en su tarea de clasificar las imágenes para identificar al edicio qie se corresponde, además que el uso de *expo* facilita en gran manera la ejecución de la aplicación en un dispositivo móvil.

Con la finalización de esta investigación, se logró obtener un software capaz de reconocer edificios mediante un modelo de inteligencia artificial. Dicha herramienta presenta grandes oportunidades de continuar con su desarrollo, debido a su escalabilidad para ser capaz de reconocer más edificios, devolver más información detallada o implementar nuevos módulos para hacer más completa la aplicación. Se creó una plataforma capaz de almacenar información acerca de lugares en la Universidad, esta información podría ser utilizada en proyectos de realidad aumentada, donde se le proporcione al usuario información acerca de lo que está viendo en todo momento.

REFERENCIAS

[1] "Recorrido Virtual." [Online]. Available: http://www.comitedeanalisis.unam.mx/recorrido-virtual.html [Accessed: 26- Apr- 2021].

[2] "Recorridos Virtuales." [Online]. Available: https://www.iberopuebla.mx/la-ibero/campus/recorridos-virtuales [Accessed: 26- Apr- 2021].

[3] "Recorrido Virtual." [Online]. Available: https://www.uic.mx/uic-360/virtualtour.html [Accessed: 26- Apr- 2021].

[4] "Visitas guiadas y excursiones en todo el mundo." [Online]. Available: https://www.civitatis.com/es/ [Accessed: 26- Apr- 2021].

[5] J. L. del Val Román, "Industria 4.0: la transformación digital de la industria," *Facultad de Ingeniería de la Universidad de Deusto*, 2016.

[6] D. Evans, "Internet de las cosas," *Cisco Internet Business Solutions Group (IBSG)*, 2011.

[7] S. I. Mariño and C. R. Primorac, "Propuesta metodológica para desarrollo de modelos de redes neuronales artificiales supervisadas," *IJERI: International Journal of Educational Research and Innovation*, 2016.

[8] B. P. Orozco, "Inteligencia Artificial," *INCyTU*, 2018.

[9] D. Ramírez, "EL MACHINE LEARNING A TRAVÉS DE LOS TIEMPOS Y LOS APORTES A LA HUMANIDAD," *UNIVERSIDAD LIBRE SECCIONAL PEREIRA*, 2018.

[10] G. Restrepo Arteaga, "Aplicación del aprendizaje profundo ("deep learning") al procesamiento de señales digitales," Master's thesis, Universidad Autónoma de Occidente, 2015.

[11] J. Padial, "Técnicas de programación "deep learning"," *Naturaleza y Libertad. Revista de estudios interdisciplinares*, 2019.

[12] A. Serrano, E. Soria, and J. Martín, *Redes Neuronales Artificiales*. Universidad de Valencia, 2009.

[13] D. J. Matich, "Redes Neuronales: Conceptos Básicos y Aplicaciones," *Universidad Tecnológica Nacional*, 2001.

[14] "Python." [Online]. Available: https://www.python.org/ [Accessed: 26- Apr- 2021].

[15] G. Zaccone, *Getting Started with TensorFlow*. 2016.

[16] A. Marzal, I. Gracia, and P. García, *Introducción a la programación con Python 3*. Universitat Jaume I, 2014.

[17] "Anaconda." [Online]. Available: https://www.anaconda.com/ [Accessed: 02- May- 2021].

[18] V. Silaparasetty, *Deep Learning Projects Using TensorFlow 2*. Apress, 2020.

[19] "Anaconda individual edition." [Online]. Available: https://www.anaconda.com/products/individual [Accessed: 02- May- 2021].

[20] "Jupyter." [Online]. Available: https://jupyter.org/ [Accessed: 02- May- 2021].

[21] "TensorFlow." [Online]. Available: https://www.tensorflow.org/ [Accessed: 26- Apr- 2021].

[22] M. Griebel, A. Dürr, and N. Stein, "Applied image recognition: guidelines for using deeplearning models in practice," *University of Würzburg*, 2019.

[23] J.-T. Park, H.-G. Kim, and I.-Y. Moon, "The JavaScript and Web Assembly Function Analysis to Improve Performance of Web Application," *International Journal of Advanced Science and Technology*, 2018.

[24] A. Purificación, *Manual de JavaScript*. EDITORIAL CEP S.L., 2011.

[25] J. Brownlee, *Deep Learning With Python*. Machine Learning Mastery, 2016.

[26] E. Castelo, "Diseño y desarrollo de una API REST sobre la que se apoyará una red social y una aplicación Android (SOCIAL PILGRIM)," *Universidad de Vigo*, 2016.

[27] "Node.js." [Online]. Available: https://nodejs.org/es/ [Accessed: 02- May- 2021].

[28] "Express." [Online]. Available: http://expressjs.com/ [Accessed: 02- May- 2021].

[29] "Mongoose." [Online]. Available: https://mongoosejs.com/ [Accessed: 02- May- 2021].

[30] Z. Li, "NoSQL Databases," *Geographic Information Science & Technology Body of Knowledge*, 2018.

[31] MongoDB, *MongoDB Atlas*.

[32] "MongoDB." [Online]. Available: https://www.mongodb.com/ [Accessed: 26- Apr- 2021].

[33] "Flask." [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/ [Accessed: 10- May- 2021].

[34] K. Relan, "Building REST APIs with Flask," *Apress*, 2019.

[35] "Gunicorn." [Online]. Available: https://gunicorn.org/ [Accessed: 10- May- 2021].

[36] M. Grinberg, *Flask Web Development*. O'Reilly Media, 2014.

[37] "React." [Online]. Available: https://es.reactjs.org/ [Accessed: 02- May- 2021].

[38] T. Khuat, "Developing a frontend application using ReactJS and Redux," *Laurea University of Applied Sciences*, 2018.

[39] "React Native." [Online]. Available: https://reactnative.dev/ [Accessed: 02- May- 2021].

[40] W. Danielsson, "React Native application development," *Linköpings universitet*, 2016.

[41] H. Van, "Building a universal application with React and React Native," Master's thesis, Haaga-Helia University of Applied Sciences, 2020.

[42] "Heroku." [Online]. Available: https://www.heroku.com/ [Accessed: 02- May- 2021].

[43] A. Quintana and S. Cassani, "CIU TOUR: Reconocimiento de edificios y monumentos mediante aprendizaje automático," *Universidad Complutense de Madrid*, 2019.

[44] J. Farfan, L. Enciso, and J. Vargas, "Towards accurate building recognition using convolutional neural networks," 2017.