UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

MACHINE LEARNING Y REALIDAD AUMENTADA PARA EL RECONOCIMIENTO DE RECURSOS TURÍSTICOS

PRESENTADA POR:

RONALD ALBERTO RENTERIA AYQUIPA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2021

DEDICATORIA

A Maurizio, Angiolina y Hesmeralda.

AGRADECIMIENTOS

Agradezco a la Universidad Nacional del Altiplano por la oportunidad de llevar superación fuera de su ámbito geográfico.

Agradezco a mi asesor de tesis por su tiempo y su apoyo incondicional en estos tiempos difíciles.

Agradezco a mis jurados de tesis por sus valiosos aportes para el mejoramiento de la presente investigación.

A mi esposa por su constante motivación e impulso para la realización de la presente tesis.

ÍNDICE GENERAL

	Pág.
DEDICATORIA	3
AGRADECIMIENTOS	4
ÍNDICE GENERAL	5
ÍNDICE DE TABLAS	i
ÍNDICE DE FIGURAS	ii
ÍNDICE DE ANEXOS	iv
RESUMEN	v
ABSTRACT	vi
INTRODUCCIÓN	1

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1.	Marco teórico	3
1.2.	Antecedentes	14

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1	Identificación del problema	23
2.2	Enunciados del problema	24
2.3	Justificación	25
2.4	Objetivos	25
2.4.1	Objetivo general	25
2.4.2	Objetivos específicos	25
2.5	Hipótesis	26
2.5.1	Hipótesis general	26
2.5.2	Hipótesis específicas	26

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1	Lugar de estudio	27
3.2	Población	28

3.3 Muestra 28
3.4 Método de investigación 29
3.5 Descripción detallada de métodos por objetivos específicos 30

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

CONCLUSIONES 83
RECOMENDACIONES 84
BIBLIOGRAFÍA 85
ANEXOS 92

Puno, 28 de mayo de 2021.

ÁREA: Ciencias de la Computación

TEMA: Machine Learning y Realidad Aumentada

LÍNEA: Sistemas, Computación e Informática

ÍNDICE DE TABLAS

		Pág.
1.	Recursos considerados como puntos de interés turístico	29
2.	Mediciones de la precisión del modelo	44
3.	Resultados de la prueba de hipótesis	45
4.	Observaciones de aciertos en la identificación de puntos de interés turístico	49
5.	Resultados de la prueba de hipótesis	50
6.	Roles del equipo Scrum del sistema de reconocimiento	53
7.	Product BackLog del sistema de reconocimiento	54
8.	Roles del equipo Scrum del sistema de gestión de contenido	63
9.	Product BackLog del sistema de gestión de contenido	63
10.	Roles del equipo Scrum de la aplicación móvil de cliente final	72
11.	Product BackLog de la aplicación móvil de cliente final	72

ÍNDICE DE FIGURAS

		Pág.
1.	Modelo de aprendizaje profundo	4
2.	Aprendizaje supervisado	5
3.	Estructura general de una red neuronal artificial	7
4.	Capas de activación y agrupación en la arquitectura CNN	10
5.	Lugar de estudio de la investigación	27
6.	Vista de recursos turísticos inventariados en Apurímac.	28
7.	Muestra del archivo train.txt	31
8.	Archivo de configuración de YOLOv3 modificado	32
9.	Página oficial de YOLOv3	33
10.	Configuración de copia de seguridad y sincronización con Colab	33
11.	Archivo obj.data que define clases y rutas	34
12.	Configuración de Colab para desplegar Darknet	34
13.	Fórmula de Haversine	35
14.	Etiquetado de recursos turísticos	39
15.	Estructura de carpetas para el entrenamiento en Google Drive	39
16.	Comando para entrenamiento por primera vez	41
17.	Comando para continuar con el entrenamiento desde un punto anterior	42
18.	Comando para validar el modelo	42
19.	Resultados de la precisión obtenida en el entrenamiento en Darknet	43
20.	Fórmula de Haversine utilizada	46
21.	Fórmula para el cálculo de la orientación inicial o rumbo	47
22.	Indicadores de posición de puntos de interés	48
23.	Información de puntos de interés turístico	48
24.	Arquitectura lógica de la implementación de los sistemas	52
25.	Organización de archivos y carpetas en Google Drive	61
26.	Resultados de validación del modelo	61
27.	Implementación de sistema de reconocimiento	62
28.	Prueba de la API de reconocimiento	62
29.	Modelo relacional de la base de datos	67
30.	Estructura de carpetas Laravel	68
31.	Capturas de pantalla gestión de usuarios	68

32. Capturas de pantalla de autenticación y sesión 69
33. Capturas de pantalla gestión de categorías 69
34. Capturas de pantalla gestión de recursos turísticos 70
35. Capturas de pantalla gestión de puntos de interés turístico 71
36. Interfaz de la app móvil 76
37. Lista de recursos turísticos entrenados 77
38. Indicación de objeto reconocido 77
39. Muestra de texto mediante realidad aumentada 78
40. Muestra de video mediante realidad aumentada 78
41. Configuración del radio de búsqueda 79
42. Localización de puntos de interés 79
43. Despliegue de información mediante realidad aumentada 80
44. App móvil instalada 80

ÍNDICE DE ANEXOS

Pág.

1. Notebook de Colab para el entrenamiento 93

2. Código Python de reconocimiento (objectRecognition.py) 94

3. Código JavaScript para la identificación de puntos de interés en Ionic 97

RESUMEN

Apurímac, a pesar de contar con gran cantidad de recursos turísticos, no ha podido difundirlos de manera adecuada, por lo que en esta investigación se pretende aplicar *machine learning* y realidad aumentada para la detección y geolocalización de recursos turísticos. Construyendo una aplicación móvil que integre todas estas tecnologías y permita mejorar la experiencia del visitante en tiempo real. Para lograr el objetivo, se consideraron 25 recursos turísticos de la región, 5 para el entrenamiento del modelo *machine learning* y 20 para la ubicación en tiempo real por geolocalización. En cuanto a *machine learning*, se entrenó con un *dataset* construido exclusivamente para esta investigación, mediante YOLOv3 sobre Darknet, a continuación, el modelo entrenado se incluyó en un servidor web con Flask sobre Python, que estará a la espera de imágenes. Además, se implementó una aplicación web para la gestión de recursos turísticos que serán mostrados al usuario final. En lo referente a realidad aumentada esta se implementó sobre una aplicación móvil la cual envía imágenes captadas por la cámara del móvil al detector, esta app móvil también permite mostrar puntos de interés cercanos basado en la geolocalización y orientación actual; ya sean reconocidos o geolocalizados, la app permite mostrar la información del recurso turístico mediante realidad aumentada. Como resultados se logró una precisión del modelo en el reconocimiento de imágenes superior al 90%, se logró determinar los puntos de interés turístico cercanos al móvil basándose en su geoposicionamiento y orientación, finalmente, se logró definir una arquitectura que intercomunique estos tres sistemas que trabajan con tecnologías diferentes.

Palabras clave: Machine learning, realidad aumentada, reconocimiento de objetos, red neuronal convolucional, tecnologías para el turismo.

ABSTRACT

Despite having a significant number of tourist resources, Apurímac has been unable to disseminate them adequately, therefore, this research aims to apply machine learning and augmented reality for the detection and geopositioning of tourist resources. It was done by building a mobile application that integrates all these technologies and improves the visitor's experience in real-time. To achieve the objective, 25 tourism resources in the region were considered, 5 for training the machine learning model and 20 for real-time location by geopositioning. Regarding machine learning, it was trained with a dataset built exclusively for this research, using YOLOv3 on Darknet, then the trained model was included in a web server with Flask on Python, which will be waiting for images. In addition, a web application was implemented for the management of tourist resources that will be shown to the end-user. This was implemented on a mobile application that sends images captured by the mobile camera to the detector regarding augmented reality. This mobile app also allows showing nearby points of interest based on geopositioning and current orientation; whether they are recognized or geolocated, the app allows the information of the tourist resource to be displayed through augmented reality. As a result, the precision of the model was achieved in the recognition of images higher than 90%. It was possible to determine the points of tourist interest near the mobile based on their geopositioning and orientation. Finally, it was possible to define an architecture that intercommunicates these three systems that work with different technologies.

Keywords: Machine learning, augmented reality, object recognition, convolutional neural network, technologies for tourism.

INTRODUCCIÓN

Apurímac es un departamento que, a pesar de contar con gran cantidad de recursos turísticos, no ha podido ponerlos en valor adecuadamente, según DDCA (2018) uno de los principales atractivos de la provincia de Abancay, el monolito de Saywite sólo obtuvo 5705 visitas durante el 2018, esto es muy poco si lo comparamos con algunos de los recursos turísticos de los departamentos vecinos.

La aplicación de tecnologías de la información en el turismo repercute positivamente en todos sus procesos, hecho que queda demostrado por diversos estudios previos. Los teléfonos inteligentes o *smartphones* han pasado de ser un simple instrumentos de comunicación a convertirse en una ventana hacia al mundo, permitiendo a los usuarios todo tipo de posibilidades. Dentro de ese mundo de posibilidades, existen aplicaciones para mejorar la experiencia turística que se utilizan en muchos lugares del mundo con gran atracción turística, pero que en Apurímac no se utilizan en la difusión de sus recursos turísticos.

El objetivo del estudio fue aplicar tecnologías de *machine learning* y realidad aumentada para el reconocimiento de recursos turísticos en la región de Apurímac. Debido a esto se logró realizar una aplicación móvil que muestra información al visitante sobre el recurso turístico, en tiempo real mediante realidad aumentada.

Para lograr lo propuesto, en lugar de utilizar herramientas comerciales de alto nivel como Wikitude, KudanAR o Vuforia, se emplearon métodos de *machine learning* para el reconocimiento de los recursos turísticos. Específicamente se utilizó YOLOv3 que es una red neuronal convolucional, la cual fue entrenada con un *dataset* o conjunto de datos propio que contiene una muestra de los recursos turísticos de Apurímac, de los cuales se capturaron las imágenes y se procesaron para formar el *dataset*. La red neuronal entrenada generó un detector, el cual se introdujo en un sistema hecho en Python alojado en un servidor web que está continuamente a la espera de imágenes, las cuales son enviadas desde una aplicación móvil para dispositivos Android, en la que el turista deberá apuntar con la cámara del móvil para que se envíen las imágenes y el detector las reconozca y envíe la información sobre el recurso turístico, la cual será mostrada mediante realidad aumentada.

Este documento se estructura de tal manera que en el capítulo I se expongan una serie de conceptos necesarios para el entendimiento del estudio, los cuales se explican

detalladamente; así como se listan un conjunto de estudios previos de los cuales se ha tomado referencia para guiar la presente tesis. En el capítulo II se identifica el problema que ha motivado el estudio, su importancia, los objetivos que se pretende alcanzar y las respuestas posibles al problema detectado. El capítulo III muestra los recursos utilizados, así como el proceso de investigación, luego se detallan los resultados obtenidos y la discusión en el capítulo IV, para finalmente plasmar las conclusiones a las que se arribó en el estudio y las recomendaciones futuras.

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1. Marco teórico

1.1.1. Machine Learning

La habilidad de aprender es el atributo más elemental del comportamiento inteligente, esta premisa abre paso al estudio y entendimiento de esta disciplina que comprende campos como ciencia cognitiva, inteligencia artificial, ciencias de la información, patrones de reconocimiento, psicología, educación, filosofía, entre otras (Kaufmann, 1983). Se plantea entonces, la creación del aprendizaje por parte de las computadoras, es decir, un aprendizaje automático, por ello que la palabra *machine learning* del inglés, tendría su traducción como aprendizaje de máquina, o de computadora y es una de las ramas de la inteligencia artificial (Allwin *et al.*, 2019).

De acuerdo a Lawson *et al.* (2019), el aprendizaje automático es una subdisciplina de la inteligencia artificial (IA), que intenta emular cómo un cerebro humano comprende e interactúa con el mundo.

El aprendizaje automático es el estudio de algoritmos informáticos que buscan mejorar automáticamente a través de la experiencia (es decir, el aprendizaje), a menudo mediante el entrenamiento en ejemplos supervisados, es decir que se retroalimentan después de una supervisión para corregir posibles fallas que tiene como objetivo, predecir la respuesta, en lugar de producir una comprensión mecánica. De hecho, el algoritmo que vincula la entrada y la respuesta no pretende representar una comprensión mecanicista de los procesos subyacentes.

En la Figura 1, se presenta el funcionamiento conceptual de un modelo neuronal, donde una persona es una combinación de ciertos atributos como bordes, esquinas

y contornos. Existe una primera capa visible (que contiene variables visibles u observables) que contiene el conjunto de datos de entrada, en el caso de una imagen se refiere a los píxeles que conforman la imagen. Seguidamente se encuentran las capas ocultas porque las variables ya no son visibles, estas capas almacenan conceptos abstractos de la imagen. Goodfellow *et al.* (2016), indican que, al visualizar el contenido de las capas ocultas, se logra encontrar los patrones. Cada una de las capas ocultas almacena datos difusos como bordes, esquinas representativas, las cuales dan una salida que es la descripción del objeto, resultado de los patrones que contiene.

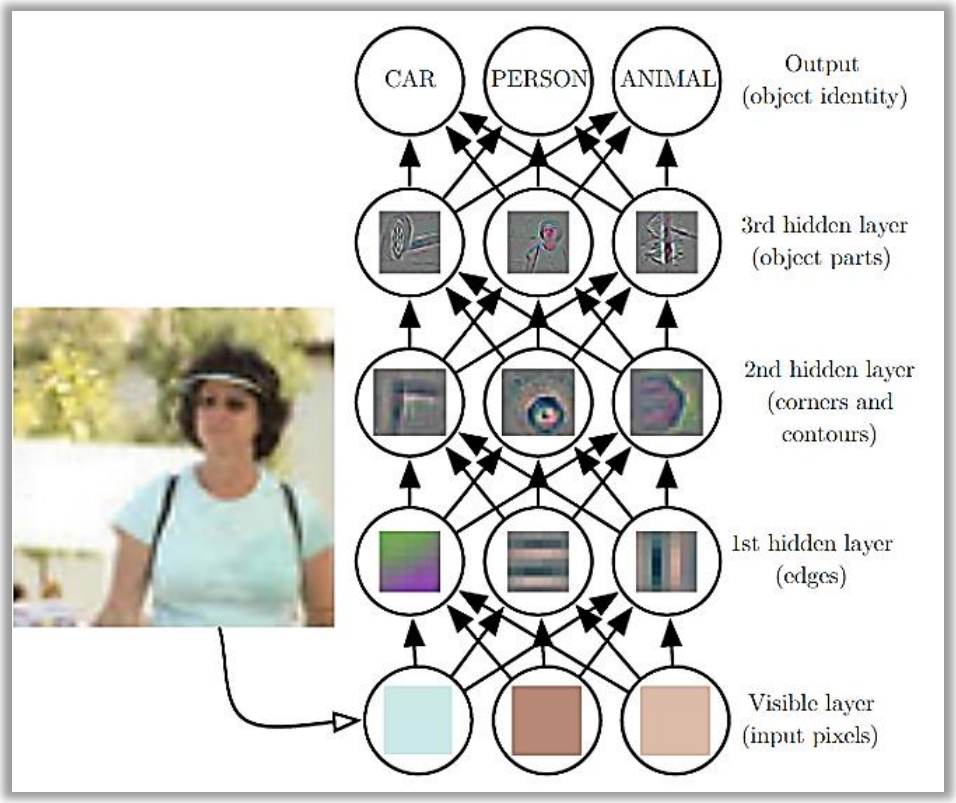


Figura 1. Modelo de aprendizaje profundo
Fuente: Goodfellow *et al.* (2016)

Tipos de aprendizaje de Machine Learning

a) Aprendizaje automático supervisado

Dado que el modelado de *machine learning* se utiliza para predecir un resultado, es necesario "supervisarlo", para observar si los valores que produce durante el entrenamiento se comparan con algunos valores objetivo, de este modo el aprendiz recibe retroalimentación. La variable dependiente se puede considerar como un profesor que supervisa al aprendiz automático mientras asigna etiquetas o

salidas numéricas a las entradas que recibe. Cuando el alumno comete un error, el maestro lo señala y le da instrucciones sobre cuál debería haber sido la etiqueta. Actualmente, los métodos de aprendizaje clásicos comúnmente utilizados incluyen BN, SVN, KNN, etc. Debido a que todo el proceso de aprendizaje tiene un propósito, el proceso de aprendizaje automático presenta una cierta regularidad y el contenido de aprendizaje es más sistemático (Jin, 2020).

Lo importante en el desarrollo de *machine learning* es elegir datos de muestra o de entrenamiento. Si los datos de un entrenamiento no pueden representar la población de datos, entonces el resultado del modelo del entrenamiento no es bueno. Por lo tanto, suele haber datos de prueba como balance. La máquina se entrena usando los datos de entrenamiento, luego el resultado se prueba usando los datos de prueba (Allwin *et al.*, 2019).

Figura 2. Aprendizaje supervisado
Adaptado de Allwin *et al.* (2019)

En la Figura 2 se muestra al maestro escribiendo los números 8, 6 y 2 e indicando la forma correcta de leerlos, donde la entrada está representada por los números y la salida es la forma de leerlos. Este conjunto de datos de entrada y salida se denominan como datos de entrenamiento. Cuando los datos se clasifican en solo dos opciones, se llama clasificación binaria; mientras que más de dos categorías se llama clasificación de etiquetas múltiples.

b) Aprendizaje automático no supervisado

El aprendizaje automático no supervisado, por otro lado, no se guía por tales comentarios, el aprendiz sin supervisión organiza los datos sin una guía explícita, (Cox *et al.*, 2020). Esto significa que el proceso de aprendizaje se confía a la propia máquina para que aprenda los conceptos y contenidos básicos y darle la libertad suficiente para completar su propio conocimiento (Jin, 2020), de esta

forma la red descubre dentro de los datos de entrada las características, regularidades, correlaciones y categorías automáticamente, esto debido a que debe tener un nivel básico de auto-organización. Las aplicaciones del aprendizaje automático no supervisado son generalmente para solucionar problemas de prototipado, agrupamiento, definición de componentes principales, familiaridad y relación entre las características comunes.

1.1.2. Red neuronal

Una red neuronal es un conjunto interconectado de elementos, unidades o nodos de procesamiento simples, cuya funcionalidad se basa libremente en la neurona animal. La capacidad de procesamiento de la red se almacena en las fuerzas o pesos de conexión entre unidades, obtenidos mediante un proceso de adaptación o aprendizaje de un conjunto de patrones de entrenamiento (Gurney, 1997).

1.1.3. Artificial Neural Network (ANN)

Una red neuronal artificial es un componente de inteligencia artificial que pretende simular el comportamiento del cerebro humano, esta red, está compuesta por un gran número de neuronas o perceptrones, donde cada neurona puede tomar decisiones simples y transmitirlas a otras neuronas organizadas en capas interconectadas (Missinglink.ai, 2018), estas decisiones se toman en base a unidades de entrada: estructuras de información basadas en un sistema de ponderación interno, es decir, que cada entrada que ingresa a la red neuronal, no necesariamente tiene el mismo peso o la misma influencia para producir la salida.

Una ANN pasa inicialmente por una fase de entrenamiento en la que aprende a reconocer patrones en los datos, ya sea visual, auditiva o textualmente. Durante esta fase supervisada, la red compara su salida real producida con lo que estaba destinado a producir: la salida deseada. La diferencia entre ambos resultados se ajusta mediante retropropagación (*backpropagation*). Esto significa que la red funciona al revés, yendo de la unidad de salida a las unidades de entrada para ajustar el peso de sus conexiones entre las unidades hasta que la diferencia entre el resultado real y el deseado produce el menor error posible.

Estructura general de una red neuronal artificial

Una red neuronal artificial se compone haciendo una similitud a las redes de neuronas, para ello, utiliza el concepto de nodo, el cual representa a una neurona

artificial. En la Figura 3, se aprecian sus componentes: La capa de entrada, es la que recepciona las señales de entrada, las cuales pueden provenir de diversos sensores o sistemas de procesamiento de bajo nivel. Seguidamente las capas ocultas extraen atributos abstractos, que son los que finalmente permitirán describir y crear patrones sobre el objeto de entrada. La capa de salida tiene la función de realizar la clasificación, interpretación o detección de la información de entrada, indicando cuál es el objeto que se corresponde con la data ingresada.

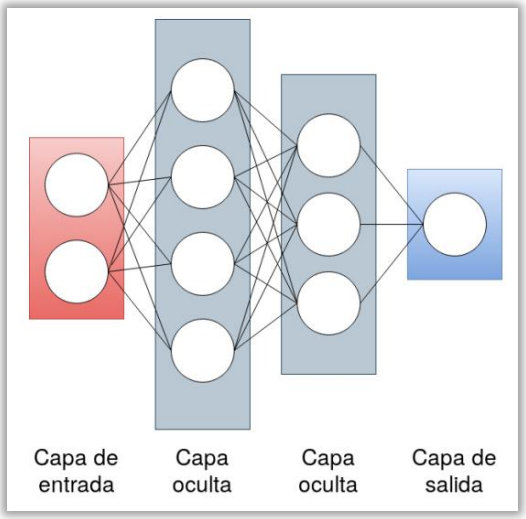


Figura 3. Estructura general de una red neuronal artificial

Aplicaciones prácticas de las ANN

Existen aplicaciones diversas de las ANN, algunos ejemplos de ellos son:

- Clasificación de texto y categorización: para búsquedas en la web, filtro de información, identificación de lenguaje, evaluación de legibilidad y análisis de sentimientos (Kim, 2017), (X. Wang & Yang, 2020), (Wangchuk *et al.*, 2020).
- Reconocimiento de entidad nombrada: es clasificar a entidades (Londres, Jorge Luis Borges, Yahoo) en categorías predefinidas que pueden ser personas, empresas u organizaciones, ubicaciones, hora, fechas, etc. (Lample *et al.*, 2016).
- Reconocimiento de voz: tiene muchas aplicaciones como en la domótica, telefonía móvil, asistencia virtual, informática manos libres, videojuegos, etc. Las redes neutrales se utilizan ampliamente en esta área (Abdel-Hamid *et al.*, 2014).
- Etiquetado de la voz: tiene muchas aplicaciones que incluyen análisis, conversión de texto a voz, extracción de información, etc. (P. Wang *et al.*, 2015).

- Reconocimiento de caracteres: se usa para el reconocimiento de caracteres de recibos, facturas, cheques, documentos de facturación legal, etc. Todos ellos, en los que se usa el reconocimiento de caracteres escritos a mano (Sharma & Dipti, 2013).
- Corrección ortográfica: permite comprobar si un texto contiene errores ortográficos. Las redes neuronales ahora se incorporan en muchas herramientas de corrección ortográfica y superan muchas de las deficiencias que tienen los métodos tradicionales de corrección ortográfica (Garaas *et al.*, 2007).
- Uso de chatbots, asistentes virtuales, detectar y eliminar el spam de la bandeja de entrada de los correos de un usuario, en el área empresarial se usa para pronosticar la dirección de las acciones de una empresa, para la calificación crediticia, en el comercio electrónico se usa para personalizar sus recomendaciones para su audiencia, para predecir la probabilidad de un evento, entre otras aplicaciones.

1.1.4. Convolutional Neural Network (CNN)

Son un tipo de redes neuronales, que constan de capas ocultas antes de llegar al proceso de salida. En estas capas, se aplican pesos diferentes a cada neurona y así definir un filtro convolucional aplicado a la información de entrada para aplicar la función de activación al resultado de la convolución (Arnal, 2018). La convolución es un "procesado distintivo" que se realiza en estas capas ocultas, y consiste en aplicar filtros al conjunto de datos de entrada para generar un mapa de características, de este modo, las convoluciones son capaces de detectar en una primera capa característica primitivas y a medida que se va profundizando en las capas, las características van encontrando detalles que configuran de forma más precisa el objeto, de esta forma se genera una nueva matriz de salida, la cual es la nueva capa de neuronas ocultas hasta finalizar con un resultado en la capa final de salida (Cruña, 2019).

En la Figura 4, se presenta una red neuronal convolucional (CNN), aplicada al reconocimiento de imágenes por medio de los procesos de convolución y agrupación, para extraer sus características esenciales, de este modo comprenderlas y así clasificar una imagen.

Los bloques de construcción básicos de CNN son:

- **Capa de convolución (convolution):** un "filtro", a veces llamado "núcleo", se pasa sobre la imagen, viendo unos pocos píxeles a la vez (por ejemplo, 3x3 o 5x5). La operación de convolución es un producto escalar de los valores de píxeles originales con pesos definidos en el filtro. Los resultados se resumen en un número que representa todos los píxeles que observó el filtro.

  "Una convolución es una operación matemática que transforma dos funciones en una tercera, la cual representa la magnitud de cuánto se superpone una función sobre la otra. Siendo f la función de entrada y g el núcleo, la convolución de f y g (el mapa de características), denotada como f $*$ g, se define como la integral del producto de ambas funciones, después de desplazar una de ellas una distancia t" (Silva, 2020).

$$(f * g)(t) = \int_{-\infty}^{\infty} f(n)g(t-n)dn$$

- **Capa de activación:** la capa de convolución genera una matriz de tamaño mucho más pequeño que la imagen original. Esta matriz se ejecuta a través de una capa de activación, que introduce la no linealidad para permitir que la red se entrene a sí misma mediante retropropagación. La función de activación suele ser ReLu.
- **Capa de agrupación (pooling):** la "agrupación" es el proceso de reducir la resolución y reducir el tamaño de la matriz. Se pasa un filtro sobre los resultados de la capa anterior y selecciona un número de cada grupo de valores (normalmente el máximo, esto se llama agrupación máxima). Esto permite que la red se entrene mucho más rápido, enfocándose en la información más importante en cada característica de la imagen.
- **Capa completamente conectada (fully connected):** una estructura de perceptrón multicapa tradicional. Su entrada es un vector unidimensional que representa la salida de las capas anteriores. Su salida es una lista de probabilidades para diferentes etiquetas posibles adjuntas a la imagen (por ejemplo, perro, gato, pájaro). La etiqueta que recibe la mayor probabilidad es la decisión de clasificación.

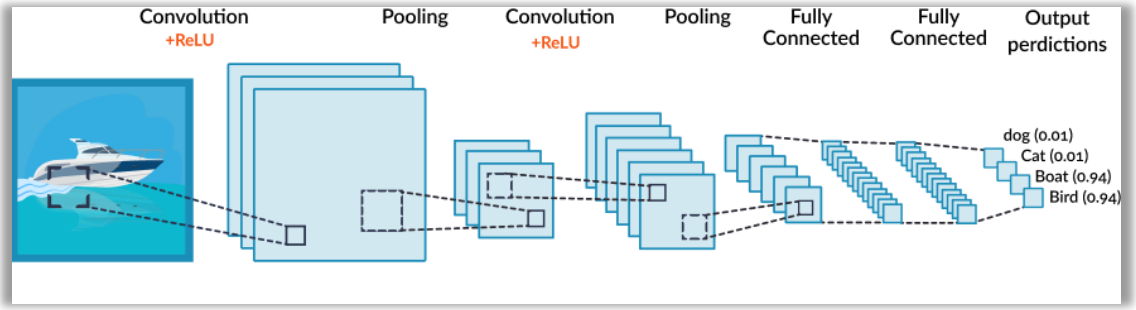


Figura 4. Capas de activación y agrupación en la arquitectura CNN
Adaptado de Missinglink.ai (2018)

Si bien, este es un ejemplo básico, puede haber múltiples capas de activación y agrupación, según la arquitectura de CNN que se plantee en cada proyecto a desarrollar.

Aplicaciones comunes para CNN para visión artificial

A continuación, se muestran algunas aplicaciones comunes de la visión por computadora impulsadas por redes neuronales convolucionales:

- **Agricultura**: los agricultores utilizan sensores hiperespectrales o multiespectrales para tomar fotografías de los cultivos y analizar las imágenes con visión por computadora para determinar su salud o la viabilidad de las semillas que se van a sembrar.
- **Automóviles autónomos:** las CNN se utilizan para la detección y clasificación de objetos, realizadas en tiempo real frente a imágenes de video en vivo de las cámaras de los automóviles. Los coches autónomos de hoy en día son capaces de identificar otros vehículos, personas y obstáculos y sortearlos con una precisión sorprendente.
- **Vigilancia**: los sistemas de seguridad modernos con capacidad de visión por computadora pueden identificar delitos, violencia o robo en secuencias de video en tiempo real y alertar al personal de seguridad. Nuevamente, esto aprovecha la detección y clasificación de objetos basada en CNN en cuadros de video.
- **Atención médica:** la visión por computadora en la atención médica ayuda a diagnosticar enfermedades como la neumonía, la diabetes y el cáncer de mama. En muchos casos, el análisis y el diagnóstico de imágenes médicas basados en CNN pueden ser tan precisos o incluso más precisos que los de un técnico o médico humano.

1.1.5. Accuracy

Es la precisión que logra un algoritmo de clasificación de aprendizaje automático, es una forma de medir la frecuencia con la que el algoritmo clasifica un punto de datos correctamente. Se suele expresar porcentualmente.

1.1.6. YOLOv3

YOLO (*You Only Look Once*) es una arquitectura de red neuronal convolucional (CNN) que es capaz de clasificar y localizar objetos en una imagen (Moreno, 2019).

1.1.7. Darknet

Darknet es un framework para el desarrollo de redes neuronales, este marco de trabajo fue escrito en C y CUDA. Se caracteriza por su facilidad de instalación y uso, además permite trabajar con CPU y GPU (Moreno, 2019). En esta investigación se utilizó este framework para implementar la arquitectura de la red neuronal YOLOv3.

1.1.8. Realidad aumentada (RA)

La realidad aumentada es una tecnología digital de vanguardia que surgió a principios de la década de 1990 (Qiu *et al.*, 2019). El concepto fue propuesto por Tom Caudell de Boeing y sus colegas cuando diseñaron un sistema de cableado auxiliar de aeronave (Dini & Mura, 2015). Azuma (1997), dividió la realidad aumentada en tres componentes importantes: fusión de realidad virtual, interacción en tiempo real y registro 3D, y postuló su aplicación potencial en muchos campos como el tratamiento médico, el entretenimiento y la industria. Como respuesta, la industria, ha venido realizando investigaciones a profundidad desarrollando dispositivos hardware auxiliares para la RA como HoloLens, MagicLeap, Epson Moverio, entre los más resaltantes; lo cual ha promovido su popularización, siendo hoy muy accesibles para el público.

La realidad aumentada es una versión mejorada del mundo físico real que se logra mediante el uso de elementos visuales digitales, sonido u otros estímulos sensoriales transmitidos a través de la tecnología, esto implica superponer información visual, auditiva u otra información sensorial en el mundo para mejorar la experiencia. A diferencia de la realidad virtual, que crea su propio entorno cibernético, la realidad aumentada se suma al mundo existente tal como es (Craig, 2013).

Breen (2020), define a la RA como “el mundo real con una capa adicional de contenido virtual (2D / 3D)”, por lo cual, realidad aumentada literalmente significa "aumentar su realidad", combinando el mundo físico con superposición de elementos virtuales generados por computadora. Estos contenidos virtuales en 2D o 3D se proyectan en realidad dentro del campo de visión de las personas (a través de la cámara de un teléfono inteligente o lentes inteligentes), con la intención de hacer que el mundo físico y virtual se reconozcan e interactúen juntos con la ayuda de la visión por computadora y el aprendizaje automático. También es la razón por la que la RA es técnicamente un gran desafío: el mundo virtual y el mundo real deben coexistir perfectamente, y la información digital debe reconocer el mundo real, con sus obstáculos, objetos naturales, edificios o gestos de personas, rostros, etc. interacciones con él, todo ello en tiempo real.

Principales aplicaciones de la Realidad Aumentada

A continuación, se presenta una visión general de las aplicaciones más comunes de la RA, aunque no se limita a ellas.

– **Medicina:** La primera aplicación real de la RA en medicina se remonta a 1986, cuando se propuso un sistema para integrar datos de tomografía computarizada en un microscopio quirúrgico (Manuri & Sanna, 2016). En la actualidad, con el avance del tratamiento de imágenes médicas, la RA pasa a ser un soporte en cirugía, así como en el diagnóstico de datos preoperatorios e intraoperatorios o en tareas de entrenamiento. Sin embargo, mucho más que otros campos de aplicación, la RA en medicina tiene que superar tres problemas principales: precisión de seguimiento, percepción errónea e interacción con datos sintéticos. La precisión requerida para varias operaciones quirúrgicas es de orden submilimétrico, por lo que los activos deben superponerse con mucha precisión. Por otro lado, la RA médica generalmente implica volúmenes interiores de trabajo muy limitados y controlados y los sistemas de seguimiento de corriente pueden proporcionar la precisión requerida en estas condiciones. La percepción errónea está básicamente relacionada con una percepción errónea de la profundidad (aunque los activos están correctamente alineados, el usuario los percibe en una posición incorrecta), pero este problema se puede mitigar utilizando dispositivos de visualización estereoscópica. El problema de la interacción se relaciona más generalmente con los problemas de diseño de la interfaz de usuario; por

ejemplo, un cirujano no puede interactuar con los activos mediante el tacto, por lo que deben implementarse interfaces de usuario naturales y multimodales. Las interfaces multimodales permiten al usuario elegir entre diferentes modos de entrada: el reconocimiento de gestos/poses y el reconocimiento de voz pueden ser dos alternativas para interactuar de forma natural con los contenidos generados por computadora.

- **Montaje, mantenimiento y reparación:** La RA, aplicada en labores complejas de mantenimiento y reparación es uno de sus campos de aplicación más prometedores, en estas tareas, que conllevan a una alta carga cognitiva derivada del continuo cambio de atención entre el dispositivo en mantenimiento y el manual, es muy útil usar RA para que se pueda manipular el dispositivo a mantener o ensamblar mientras se recibe asesoría técnica ya sea de otro dispositivo por audio o también con el apoyo de expertos remotos que dan soporte mientras se realizan los procedimientos técnicos, es decir, combinados con lo tele-presencial (Vorraber *et al.*, 2020).
- **Entretenimiento, deporte y marketing:** En la industria del entretenimiento la realidad aumentada permite a los participantes ser parte del juego, involucrándolos de forma más cercana. La RA tiene como objetivo cerrar la brecha entre lo real y lo virtual, donde el mundo real puede convertirse en el escenario de un juego y el jugador puede experimentar una modalidad de juego completamente nueva y emocionante, por lo tanto, es la mejor herramienta para brindar a los usuarios una nueva experiencia de juego. Sumado a ello, las consolas de juegos modernas también implementan RA mediante el uso de diferentes tipos de cámaras para aumentar los gráficos de computadora en imágenes en vivo (Cavallaro *et al.*, 2011).
- **Turismo:** Con la mejora de los teléfonos inteligentes y tabletas de nueva generación, que a menudo están equipados con sensores GPS y conexiones de internet rápidas, las aplicaciones de realidad aumentada para el turismo han permitido enriquecer la experiencia turística. Básicamente, se pueden clasificar tres tipos de aplicaciones de RA para el turismo. Las guías aumentadas son el primer tipo de aplicación de RA para el turismo; una guía aumentada busca, recupera y visualiza información recopilada de varias fuentes de Internet (por ejemplo, portales turísticos). La información está ordenada con el fin de brindar a los

usuarios todo el apoyo necesario para: organizar viajes, reservar tours, alquilar autos, etc.

- **Enseñanza, educación y formación:** En el campo de la educación, el uso de RA mejora las rutas tradicionales de aprendizaje y formación por medio de la interacción, cooperación y colaboración entre estudiantes como con el profesor. Se aplica al utilizar sistemas de escala microscópicos o macroscópicos; además, se pueden representar eventos peligrosos y/o destructivos de diversas disciplinas como la Física, Química, Matemáticas y Geometría, Geografía, Astronomía, Historia, Arqueología, Música y Arte por citar algunos. El uso más común de la RA en la educación está relacionado con el material del curso interactivo (que a menudo proporciona visualizaciones en 3D), lo que permite a los educadores reducir la brecha entre lo real y lo virtual.

1.2. Antecedentes

Para realizar el reconocimiento de recursos turísticos, existen trabajos previos que utilizan otras tecnologías para lograr este objetivo, tal como en la investigación de Amato *et al.* (2011), quienes realizaron una aplicación de apoyo visual al turismo interactivo en La Toscana, la cual es una guía interactiva accesibles a través de teléfonos inteligentes. La interacción del usuario se obtiene principalmente mediante el uso de imágenes. Para recibir información sobre un monumento en particular, los usuarios deben tomar una foto del mismo; es en este momento que se despliegan las técnicas de análisis de imágenes y reconocimiento de contenido para determinar automáticamente los monumentos fotografiados y la información pertinente se muestra al usuario, haciendo uso de puntos de referencia y clasificadores de imágenes.

El trabajo de Parvez (2020) fue de tipo descriptivo, y profundiza en el uso del aprendizaje automático en los servicios turísticos y organizacionales permitiéndoles innovación de alta tecnología en la industria hotelera. Los hallazgos fundamentales son que *machine learning* permite predecir el futuro de las condiciones comerciales a través de los datos recopilados, que los clientes pueden revisar y planificar toda la preparación para el uso de servicios turísticos, encontrar información. Por lo tanto, los hoteles instalan máquinas para realizar tareas específicas con precisión, como reservas y ocupación, gestión de ingresos, segmentación, previsión de la demanda, gestión de rendimientos, fijación de precios, evaluación del rendimiento, seguimiento de marca, análisis competitivo y recopilación de datos internos y externos para comprender el comportamiento del cliente. En este

segmento, el procedimiento de ML está dirigido a la operación, además la organización puede investigar nuevos datos de demanda, comportamiento y tendencia turística futura de los clientes.

Afsahhosseini & Al-mulla (2020), en su artículo "Machine learning in tourism", presenta las aplicaciones que se pueden dar al aprendizaje automático aplicado al turismo, entre las que se citan: el uso de fotografías que toman los usuarios, para aprender patrones de movimiento de los turistas, especialmente a partir de fotografías geoetiquetadas; mapas: que cuentan con geolocalización y permiten el aprendizaje del desplazamiento del turista; aprendizaje automático en el turismo, para conocer el antes, durante y después del viaje, a fin de pronosticar la demanda turística.

Chawla (2019), presenta un estudio descriptivo de siete aplicaciones exitosas de inteligencia artificial y aprendizaje automático en la industria de viajes, entre ellas se encuentran: i) Sistemas de predicción: usado para obtener el precio óptimo en la compra o reserva de un servicio o producto del sector turístico de viajes, ii) *Chatbots*, permite brindar un servicio al cliente completo las 24 horas, los 7 días de la semana, que reduce automáticamente la carga de personal para responder instantáneamente en cualquier momento del día o de la noche, brindando un apoyo, iii) Gestión de la experiencia del usuario, se evalúa usando el aprendizaje automático para evaluar los aspectos que transmite el sitio web como la interacción, la investigación, la experiencia hasta el intercambio de impresiones con familiares y amigos, para comprender al usuario, iv) Sistemas de recomendación, se usa para atraer a clientes al sitio web o aplicación de viajes, y de esta forma comprender las necesidades, el presupuesto y las preferencias de cada cliente, alimentándose con datos históricos como reservas previas, comportamiento o datos en tiempo real del viajero, v) Optimización de ventas, para apoyo en todo el proceso de ventas a través de sugerencias que se procesan de datos históricos y tendencias, vi) Optimización de costos, permite sincronizar sus estrategias de precios en tiempo real y presentar el precio correcto, adecuándose a características específicas de los usuarios, como días preferidos y tiempo de vuelo, tiempo de compra de pasajes, conveniencia, etc., vii) Detección de fraudes, para identificar comportamientos anormales y crear puntajes de riesgo para construir una comprensión completa de cada transacción de pago de un cliente y descifrar si son pagadores confiables, detectar anomalías con mayor precisión y rapidez y, por lo tanto, asegurar transacciones en línea.

En el trabajo de X. Wang *et al.* (2020), se realizó el reconocimiento y clasificación de imágenes mediante una red neuronal de convolución con 19 capas de profundidad para garantizar su alta precisión y eficiencia. El documento principalmente diseña, implementa, optimiza y ajusta la estructura del modelo de la red convolucional de la plataforma marco TensorFlow. La estructura del modelo de red neuronal de convolución diseñada consta de 8 capas de capa de convolución, 6 capas de capa de agrupación máxima, 2 capas de capa de convolución basada en Google Net Mixed1 y Mixed2, 2 capas de capa completamente conectada y 1 capa de clasificación Softmax. Los resultados experimentales muestran que los modelos de redes neuronales de convolución presentados son superiores a otros modelos de redes neuronales en precisión y eficiencia de reconocimiento y clasificación de imágenes, y tienen un buen papel rector en la resolución de problemas prácticos de ingeniería. Además, solo usa una función de activación, una función de pérdida, un clasificador y otras funciones que no se probaron en el estudio.

El trabajo de Quintero *et al.* (2018), presenta el uso de reconocimiento de imágenes basado en el uso de algoritmos de redes neuronales profundas, para lograr el aprendizaje de patrones. Se realizó el entrenamiento de una red neuronal convolucional implementada en Tensorflow y Keras, usando la arquitectura Inception v3. El objetivo es reconocer por medio de imágenes a macroinvertebrados en el agua de consumo humano en organizaciones comunitarias de Panamá, esto con el fin de garantizar el acceso y calidad del agua. Los resultados muestran que el modelo basado en Inception v3 permite aplicar el proceso de transferencia de conocimiento a la red neuronal y los resultados obtenidos en la clasificación de imágenes se encuentran por encima del 90% para el conjunto de entrenamiento y validación. En cuanto al error de clasificación se llevó hasta el 6% en el conjunto de imágenes de entrenamiento, pero no soportó el sobre-entrenamiento, esto indica que el modelo sufrió de sobreajuste, por lo cual es necesario alimentar la base de datos de imágenes.

Lou & Shi (2020) realizaron la implementación de una red neuronal convolucional para el reconocimiento de escenas, se utilizó un conjunto de datos llamado PLACES2 que contiene casi 7 millones de imágenes etiquetadas de varias escenas, este fue un conjunto de datos suficientemente exhaustivo para entrenar y probar un modelo de reconocimiento. El primer modelo, Inception v3 se entrenó en el conjunto de datos, logrando una precisión media del 89,3%, el modelo 2 y 3 se probaron en otros dos conjuntos de datos SUN397 y NUS-WIDE que contienen 9824 y 4499 imágenes respectivamente. Aquí también el

modelo Inception v3 funcionó bien, al lograr altas precisiones de predicción de 94,6% y 85,7% respectivamente, en cuanto al conjunto de datos NUS-WIDE se logró una precisión solo del 85,7%. Por lo tanto, se puede sugerir que el modelo Inception v3 CNN puede usarse para realizar propósitos de identificación y reconocimiento de escenas a partir de grandes conjuntos de imágenes.

Cobeña *et al.* (2019) realizaron un trabajo usando convolución de redes neuronales para reconocimiento y procesamiento de imágenes satelitales de cultivos de plantación de caña de azúcar en la región Troncal de la Costa del Ecuador. Las áreas geográficas fueron captadas por un satélite con diferentes ángulos y diferentes geolocalizaciones. Para reducir el margen de error, se generaron 25 épocas de entrenamiento. El uso de más capas, una mayor cantidad de imágenes y otros filtros geográficos a fin de generar predicciones mucho más precisas. La CNN se creó con solo 2 capas de procesamiento y mostró un promedio de predicción del 94,16%. En este trabajo sólo se utilizaron 840 imágenes puesto que la plataforma que las provee tiene un límite de descarga por día y no todas las imágenes son útiles, por ejemplo, no se usaron las imágenes que tienen un nivel de nubosidad superior al 22%.

Traore *et al.* (2018) usaron las redes neuronales de convolución para clasificar patógenos epidémicos como virus, bacterias, protozoos, priones, etc. El trabajo se sustenta en la falta de especialistas en el manejo de microscopios que genera una pérdida considerable de tiempo mientras que la epidemia continúa propagándose. La propuesta es hacer más inteligentes los futuros microscopios para que puedan indicar por sí mismos la existencia o no del patógeno de una epidemia en una muestra usando CNN en cinco fases de aprendizaje profundo: (1) Proporcionar imágenes de conjuntos de datos de entrenamiento al microscopio (2) Entrenamiento de la arquitectura CNN (3) Datos de prueba preparados (4) Aplicar el modelo generado por CNN en los datos de prueba y finalmente (5) Evaluar los resultados de la clasificación y luego integrar la solución en microscopios futuros. Los resultados obtenidos son de una precisión de clasificación del 94%, con 200 imágenes de *Vibrio cholerae* y 200 imágenes de *Plasmodium falciparum* para el conjunto de datos de entrenamiento y 80 imágenes para los datos de prueba.

Burkapalli & Patil (2019) realizaron una investigación de segmentación de imágenes de alimentos usando CNNs profundos en la India, el cual se puede considerar como un caso de reconocimiento visual de tipo fino, pues varias fotos de la misma categoría

generalmente tienen una variabilidad significativa, por ello se requería una técnica eficaz de segmentación y clasificación para identificar las cocinas particulares y un análisis detallado usando redes neuronales convolucionales profundas (DCNN) adaptativas de borde (EA), donde cada imagen de entrada se divide en parches para proporcionar una descripción estructural de datos mucho más eficiente y precisa. Los hallazgos encontrados muestran que: EA-DCNN comienza con el desarrollo de un mapa aproximado de características que se obtiene a través de DCNN, luego se aplica el modelo de EA para construir la imagen segmentada final, que ayudan a la red convolucional a optimizar el desempeño de la segmentación en una extensión significativa.

Lawal *et al.* (2019) lograron en su investigación de: "Análisis de reconocimiento de género basado en rostros para nigerianos que usan CNN" para realizar reconocimiento facial, que es una rama de la biometría que utiliza el rostro, que es un rasgo físico, para identificar de forma única a las personas. La metodología consistió en tener un conjunto de datos de más de 6000 imágenes de hombres y mujeres con diferentes variaciones, 3063 imágenes de rostros de hombres y 3063 de mujeres. La base de datos se dividió en datos de entrenamiento y datos de prueba, además se preprocesaron las imágenes a una misma dimensión y conversión a escala de grises. La arquitectura de CNN adoptada constó de solo 3 capas convolucionales, por lo tanto, no fue CNN profunda y fue realizada en el software MATLAB para implementar el código. Finalmente, lograron una precisión de reconocimiento general del 98,72%, lo que demuestra la viabilidad y el potencial de investigación en esa dirección, aunque ha sido trabajada solo con rostros nigerianos.

Hu *et al.* (2020) presentaron un estudio descriptivo sobre las CNN, encontrando tres características definitorias: i) En el proceso de reconocimiento de imágenes, si el tamaño del filtro es apropiado o no, tiene un impacto directo en el proceso de entrenamiento y la precisión del reconocimiento. Por lo tanto, para mejorar el resultado del reconocimiento de imagen, es necesario seleccionar el filtro con el tamaño más adecuado, ii) Cuando se utiliza CNN para el reconocimiento de imágenes, la profundidad de la red de diferentes selecciones de problemas suele ser diferente, y el valor de profundidad debe determinarse mediante preselección manual y experimentación, lo que limita la universalidad de la estructura de la red. Por lo tanto, en aplicaciones prácticas, necesitamos seleccionar una profundidad de estructura de red general aproximada para problemas específicos de reconocimiento de imágenes y iii) La aplicación de CNN en el reconocimiento de imágenes logra principalmente el efecto de aplicación a través del entrenamiento de conjuntos de

datos, que tiene grandes limitaciones para diferentes conjuntos de datos. Por lo tanto, la CNN necesita entrenar el conjunto de datos existente de acuerdo con diferentes conjuntos de problemas para obtener el mismo resultado. Si la distribución del conjunto de datos de entrenamiento es diferente de la del conjunto de datos de prueba, es difícil para la red neuronal convolucional obtener un buen resultado de reconocimiento. Esto lleva a la conclusión que, en el procesamiento de imágenes basado en CNN, no se ha formado una teoría normal completa.

Xie *et al.* (2019) proponen un modelo para reconocimiento facial utilizando el marco Python + Keras, con la base de datos Olivettifaces, su método involucra principalmente dos aspectos: uno es observar la influencia en la red cambiando el número de neuronas en la capa oculta; la otra es observar la influencia en la red cambiando el número de mapas de características de la capa convolucional 1 y la capa convolucional 2. La principal diferencia entre CNN y otras redes neuronales tradicionales es que el procesamiento de capas convolucionales y agrupadas se agregan para extraer características de la imagen. Los resultados experimentales muestran que la tasa de reconocimiento del modelo CNN en la base de datos Olivettifaces es del 97,5%. Cuando se utiliza el modelo CNN óptimo, la tasa de reconocimiento promedio es cercana al 100%, lo que verifica la validez y precisión del algoritmo y modelo.

El reconocimiento de imágenes ha sido aplicado en la investigación de Zherdev (2017), basado en el cálculo de índices de conjugación con vectores de clase, las imágenes de los objetivos forman los vectores de características que se transforman usando redes neuronales convolucionales preentrenadas. El objetivo fue reducir la complejidad computacional con la calidad de reconocimiento superior. Se obtuvo mediante un procedimiento de excepción vectorial a partir de clases y con extracción previa de características de imágenes de 64x64 píxeles. Se entrenó CNN en este conjunto de datos de imágenes y, como resultado de las convoluciones, se recibe el vector de características con 120 componentes. Por lo tanto, aumenta la complejidad computacional en la etapa de entrenamiento, pero en la etapa de prueba el algoritmo se ejecuta más rápido además de no disminuir la calidad del reconocimiento.

La investigación de Dewantoro *et al.* (2020) tuvo el objetivo de conocer la eficacia del algoritmo YOLO aplicado en la detección del número de vehículos en la carretera, usando un conjunto de datos creados por los autores y también datos de YOLO. El resultado

muestra que el algoritmo YOLO detecta con éxito vehículos hasta en un 65,3% del total de vehículos que pasan por la carretera y le da la etiqueta incorrecta hasta en un 20,7% del total de la etiqueta dada si se usa el conjunto de datos original de YOLO, sin embargo al usar el conjunto de datos propio creado por los autores, el algoritmo YOLO detecta con éxito vehículos como máximo hasta en un 9,3% del total de vehículos que pasan por la carretera y da una etiqueta incorrecta hasta en un 7,4% de la etiqueta total.

Sumit *et al.* (2020) realizaron un trabajo comparativo en el reconocimiento de figuras humanas usando YOLO y Mask R-CNN. El conjunto de datos ha sido proporcionado por los mismos autores, que contienen figuras humanas aleatorias tanto superpuestas como no superpuestas, que constan de 500 imágenes en diferentes ubicaciones geográficas con diferentes orígenes. Los resultados indican que el desempeño de YOLO ha sido superior, detectando pequeñas figuras humanas entre otras imágenes humanas prominentes con mayor precisión, además que YOLO realiza la detección en tiempo menor. Dado que los algoritmos de aprendizaje automático son en su mayoría datos específicos, los autores creen que los resultados presentados pueden variar con la naturaleza variable de los datos bajo observación. De otra manera, los datos presentados podrían verse como un contra-ejemplo de desvelar la inexactitud de detección de Mask R-CNN.

Cheng (2020) realizó una comparación entre la red neuronal convolucional (Region based-CNN) y YOLO en sus versiones 1, 2 y 3 en la identificación de imágenes. Sobre la R-CNN se indica consta de cuatro pasos, que incluyen ingresar imágenes, extraer propuestas de región, computar características de CNN y clasificar regiones. En resumen, CNN, SVM (Sector Vector Machine) y el modelo de regresión están incluidos en R-CNN. Como resultado, R-CNN es muy computacional, logrando un efecto de detección excelente, que no solo reduce el costo de tiempo, sino que también mejora la calidad de la propuesta. Su principal ventaja es que tiene una tasa de error más baja que la CNN convencional, al dividir las imágenes en regiones individuales. Además, otra ventaja es que la red CNN puede extraer características de la imagen automáticamente. Por otro lado, YOLO es una red neuronal convolucional que admite el entrenamiento y las pruebas de extremo a extremo y puede detectar y reconocer múltiples objetivos en imágenes con cierta precisión. El artículo presentó la aplicación a la detección de texto en escenas naturales con CNN y con YOLO V2, con un conjunto de datos de fabricación propio, seguidamente se entrenó el modelo, y los resultados comparativos obtenidos fueron: YOLO V2 obtiene una precisión de 73.54% y un tiempo de reconocimiento de 0.105 segundos

con el conjunto de datos creado por los autores, 76.8% con el data set VOC2007 y un tiempo de 0.014 segundos y un 67.25% con el conjunto de datos propio de YOLO y un tiempo de 0.122 segundos; CNN logra 76.29%, con un tiempo de 6.9 segundos. Como conclusión se observa que, si bien es cierto, el porcentaje de precisión no varía significativamente entre las bases de datos usadas y entre el algoritmo YOLO y CNN, existe una clara diferencia entre los tiempos de ejecución, siendo YOLO quien obtiene un desempeño muy superior a CNN.

En cuanto a proyectos que usan la realidad aumentada para el sector turismo, se tiene el de Fernández & Cuadrado (2014), con el artículo científico: El impacto de las nuevas tecnologías en el sector turístico: aplicación de la realidad aumentada al turismo cultural, dicen: La estrecha relación existente entre el turismo y la innovación tecnológica ha generado un importante abanico de posibilidades para empresas y destinos turísticos en su propósito de otorgar una experiencia más enriquecedora a los turistas. Así, el uso de las nuevas tecnologías se ha convertido en un elemento indispensable para cualquier destino turístico, más aún en un entorno en el que prevalece la participación activa del usuario con y en el destino a través de las redes sociales o las aplicaciones informáticas, entre otras herramientas que contribuyen lograr una experiencia en el cliente única y diferenciada.

Callejas *et al.* (2011), también utilizan la realidad aumentada pues consideran constituye uno de los avances tecnológicos más importantes de los últimos años, y que consiste en la fusión de un entorno real con un entorno virtual basado en la sobreimpresión de objetos virtuales sobre la realidad en tiempo real. La realidad aumentada es una eficaz y relevante herramienta de comunicación a través de la cual es posible dar a conocer los recursos turísticos, de ahí que su uso en este sector y, particularmente, su aplicación al turismo cultural se haya convertido en un valor añadido para dar a conocer el patrimonio histórico y cultural de un destino turístico. Es por ello que en este trabajo se analizaron las posibilidades derivadas del uso de la realidad aumentada en el ámbito turístico, y particularmente en su aplicación al turismo cultural, centrando la investigación en el estudio de un caso, el de la empresa Vaivén Gestión Turística y Cultural, a través de su aplicación móvil Guideo App.

Caballero & Villacorta (2014) en la tesis: Aplicación móvil basada en Realidad Aumentada para promocionar los principales atractivos turísticos y restaurantes calificados del

Centro Histórico de Lima, dicen: El presente proyecto consiste en desarrollar una aplicación móvil que permita al turista acceder a la información relevante, durante su visita, de los principales atractivos turísticos y restaurantes calificados del Centro Histórico de Lima a través de la realidad aumentada en los smartphones debido a que existe escasez de herramientas tecnológicas que permitan a los turistas acceder a dicha información a pesar que en la actualidad, los turistas, hacen uso de sus dispositivos móviles durante su viaje. El desarrollo del presente proyecto se ha llevado a cabo haciendo uso de la metodología ágil SCRUM por tratarse de un proyecto con un tiempo ajustado. El resultado obtenido fue brindar al turista una herramienta tecnológica usando la realidad aumentada en el smartphone que le permita acceder a información relevante tanto de atractivos turísticos como restaurantes y de esta manera mejorar la experiencia, de acceso a la información del turista durante su visita. Por lo que se concluye que la aplicación móvil contribuyó con la difusión de los atractivos turísticos y restaurantes del Centro Histórico de Lima a través de una aplicación móvil innovadora haciendo uso de un Smartphone cuyo beneficio del turismo interno es la de brindar información real y exacta de los principales lugares del Centro Histórico de Lima.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1 Identificación del problema

Apurímac es cuna de muchos atractivos turísticos y cuenta con un patrimonio cultural vasto compuesto por inmuebles arqueológicos e históricos a lo largo de su territorio. Para realizar la puesta en valor de sus sitios arqueológicos y turísticos, muchos países han optado por presentar soluciones digitales que proporcionan medios más atractivos, informativos e interactivos. De acuerdo a Carreton (2017), la gestión del Patrimonio Cultural tiene como objetivo proteger y difundir. En cuanto a proteger, esta labor la cubren diversos organismos del estado, por otro lado, difundir trata de llegar a la gente, de educar, explicar, dar a conocer. El objetivo es proteger. Si se conoce, se aprecia. Si se aprecia, se protege.

Por otro lado, diversos resultados obtenidos en investigaciones previas afirman que el uso de la tecnología mejora la experiencia del turista, como en el trabajo de Pierdicca *et al.* (2019), que concluyen que los servicios digitales representan el único instrumento capaz de transmitir información de forma rápida y ágil. Barrile *et al.* (2019) entienden que las nuevas tecnologías permiten mejorar la dicotomía tradición-innovación, donde el espectador no se limita a observar, sino que también utiliza otros sentidos para vivir una experiencia real para recordar, generando una vivencia en el visitante. C. Wang (2011) considera que un servicio más perfecto es la inclusión de servicios digitales que proporcionan mayor consumo para el viajero y más oportunidades para el destino de viaje.

En este contexto, existen variadas soluciones tecnológicas para enriquecer la experiencia de usuario, una de ellas es la obtención de datos del atractivo turístico a visitar en tiempo real. Para lograr esta labor en el instante mismo, es posible aplicar el uso de *machine*

learning en el reconocimiento de objetos, más específicamente con el uso de Redes Neuronales Convolucionales (CNN). Esta disciplina permite que un turista apunte con la cámara de su celular al recurso turístico y pueda reconocerlo, vinculando además con la tecnología de realidad aumentada que brinde información relevante del recurso u objeto turístico. Con este objetivo, es necesario que la CNN esté diseñada para lograr una precisión adecuada y reconocer el objeto o recurso turístico, permitiendo una experiencia que complemente el momento vivido brindando información sobre el sitio turístico visitado, además del uso de la geolocalización para determinar la posición del viajero y poder presentarle otros sitios de interés a su alrededor, todo ello haciendo uso de la realidad aumentada, los que se desplegarían finalmente en una aplicación móvil de fácil acceso y descarga.

Este tipo de tecnologías, que mejoran la experiencia turística en tiempo real, son muy escazas no se han implementado en la región de Apurímac; pero se pueden realizar utilizando herramientas comerciales de muy amplio uso como Wikitude, Vuforia, KudanAR o Layar; las cuales tienen la ventaja de permitir desarrollos rápidos y fáciles, pero, por otra parte, tienen como limitante que son poco flexibles o escalables, además que son de pago, algunas de la cuales elevado. Esto motivó a que se implementara una solución propia utilizando diversas tecnologías que coexisten y se comunican para lograr el objetivo.

Además, con la situación actual de emergencia sanitaria, producto del virus SARS-CoV-2, en la que vive el mundo, una aplicación que reconozca recursos turísticos y brinde información mediante realidad aumentada, es una manera de evitar la aglomeración de turistas para recibir la información de un guía turístico.

2.2 Enunciados del problema

La identificación del problema permitió formular las siguientes preguntas:

- ¿Cuál será el nivel de precisión del modelo entrenado en el reconocimiento de recursos turísticos de Apurímac usando Machine Learning?
- ¿Cómo determinar en tiempo real, los puntos de interés turístico cercanos, en base a la geolocalización y orientación de la cámara del móvil, a través de una app móvil con realidad aumentada?
- ¿Cómo será la arquitectura que permita la interacción entre Machine Learning, Realidad Aumentada, Aplicación Web y Aplicación Móvil en el reconocimiento de recursos turísticos de Apurímac?

2.3 Justificación

En los últimos años, la sociedad en general ha tomado mayor conciencia de la puesta en valor del patrimonio de una región y que puede hacer una significativa contribución a su desarrollo social y económico. Estos lugares de importancia cultural y turística enriquecen la vida de las personas, a menudo brindando un sentido profundo e inspirador de conexión con la comunidad y el paisaje, el pasado y las experiencias vividas. Son registros históricos, que son expresiones importantes de la identidad y experiencia humanas. Los lugares de importancia cultural y turística, reflejan la diversidad de las comunidades, hablan de quiénes somos y del pasado que nos ha formado (ICOMOS, 2013).

El patrimonio cultural expresado a través de los recursos turísticos, puede hacer una significativa contribución al desarrollo social y económico. Existe una creciente conciencia sobre la contribución potencial del patrimonio al desarrollo regional. Estos activos cuentan con un valor que se origina en el potencial económico y social que tienen de generar un flujo de servicios o de recursos una vez puestos en uso (SUBDERE, 2012). Sin embargo, al recorrer un sitio turístico muchas veces no se cuenta con un guía que indique y dé a conocer esta historia o simplemente información acerca del recurso, lo cual enriquece la visita y genera una conexión más profunda con el visitante. Por otro lado, el fuerte vínculo de la tecnología con el sector turístico puede ser aprovechado para brindar esta información, pero con la demanda de data en tiempo real se hace necesario el uso de tecnologías que permitan obtenerla en el momento mismo de la visita, facilitando el recorrido y siendo un apoyo, además de vincular con diversos tipos de formatos multimedia. Utilizando para lograr este propósito tecnologías promisorias como *machine learning*, realidad aumentada, así como aplicaciones web y para móviles.

2.4 Objetivos

En virtud a todo lo manifestado se plantearon los siguientes objetivos:

2.4.1 Objetivo general

Aplicar *Machine Learning* y Realidad Aumentada en el reconocimiento de recursos turísticos de Apurímac.

2.4.2 Objetivos específicos

- Determinar el nivel de precisión del modelo entrenado en el reconocimiento de recursos turísticos de Apurímac al usar *Machine Learning*.

- Determinar en tiempo real, los puntos de interés turístico cercanos, en base a la geolocalización y orientación de la cámara del móvil, a través de una app móvil con realidad aumentada.
- Definir una arquitectura que permita la interacción entre *Machine Learning*, Realidad Aumentada, Aplicación Web y Aplicación Móvil en el reconocimiento de recursos turísticos de Apurímac.

2.5 Hipótesis

2.5.1 Hipótesis general

Al aplicar *Machine Learning* y Realidad Aumentada se logra el reconocimiento de recursos turísticos de Apurímac.

2.5.2 Hipótesis específicas

- Al aplicar *Machine Learning*, se logra un nivel de precisión adecuado del modelo entrenado para el reconocimiento de recursos turísticos de Apurímac.
- Al aplicar técnicas de geoposicionamiento, la app móvil identifica correctamente los puntos de interés turístico cercanos en tiempo real.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 Lugar de estudio

La investigación se realizó en el año 2020, en el departamento de Apurímac, provincia de Abancay, en los distritos de Abancay, Tamburco y Curahuasi. Dentro de los cuales se consideraron diversos recursos turísticos para el estudio, destacando: i) Complejo arqueológico de Saywite, distrito de Curahuasi, ii) Parque Micaela Bastidas, distrito de Abancay, iii) Plaza de Armas de Abancay y iv) Museo Arqueológico y Antropológico de Apurímac, tal como se presenta en la Figura 5.

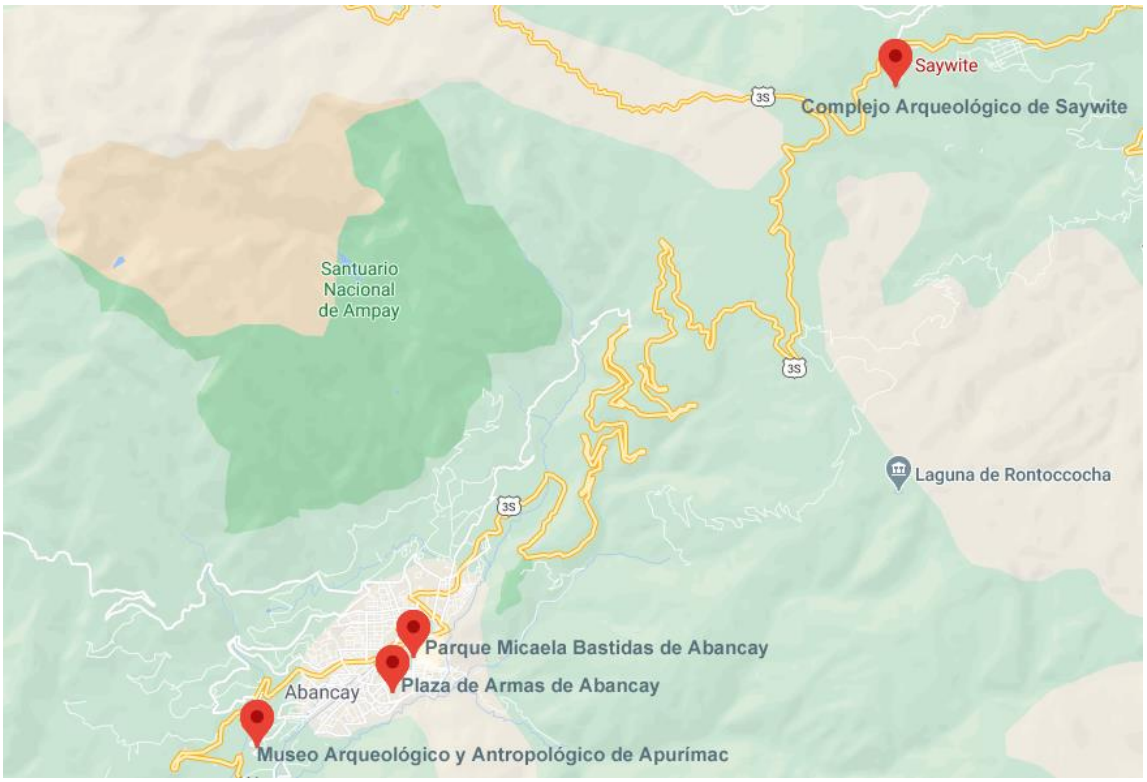


Figura 5. Lugar de estudio de la investigación
Fuente: Google Maps

3.2 Población

La población para esta investigación está compuesta por la totalidad del recurso turístico del departamento de Apurímac. Estos recursos son registrados en el Inventario Nacional de Recursos Turísticos administrados por el MINCETUR. Según SIGMincetur (2021) la cantidad de recursos turísticos inventariados del departamento de Apurímac es de 112, como se observa en la *Figura 6*. Por lo que la población del estudio es N=112.

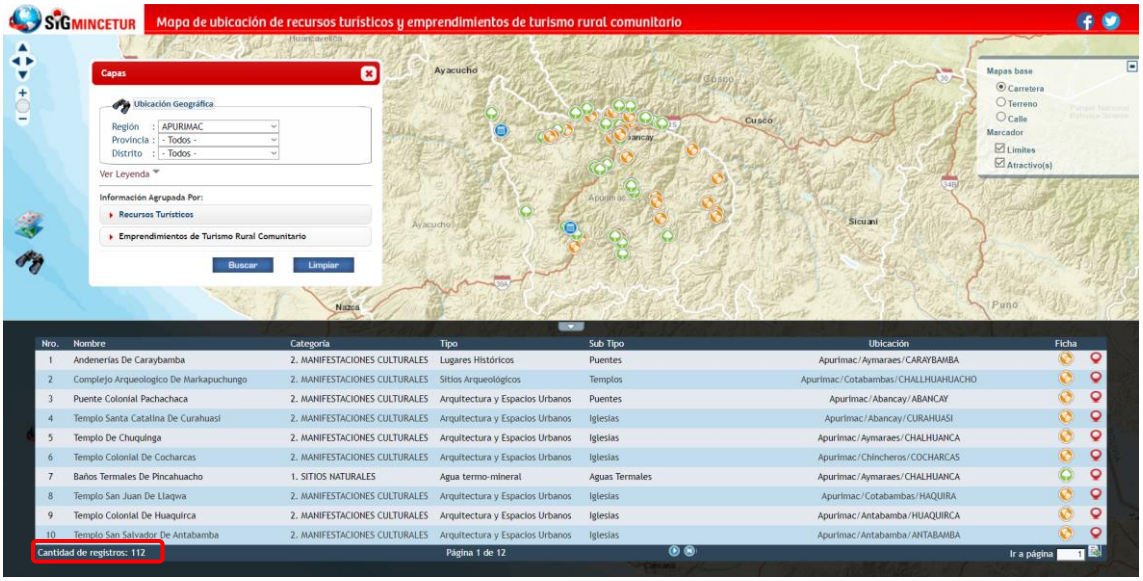


Figura 6. Vista de recursos turísticos inventariados en Apurímac.
Fuente: SIGMincetur (2021)

3.3 Muestra

Debido a la situación que se está viviendo en el mundo por la pandemia del coronavirus que ha llevado a decretar cuarentenas y restricciones de tránsito, además de considerar que la cantidad de recursos turísticos (clases en el modelo) no necesita ser tan grande para el entrenamiento, es que no es factible realizar un muestreo aleatorio.

Por lo tanto, la muestra fue no probabilística por conveniencia. Se eligieron por acceso y distancia cinco recursos turísticos ubicados en el distrito de Abancay y el distrito de Curahuasi, sobre los cuales se realizó el entrenamiento del modelo de una red neuronal convolucional para su reconocimiento, y son los siguientes: i) Monolito de Saywite, ii) Pérgola de la Plaza de Armas, iii) Estatua de Micaela Bastidas Puyucahua, en el parque del mismo nombre, iv) Momia enfardelada, en el Museo Arqueológico y Antropológico de Apurímac y v) Orejeras y nariguera de oro, en el Museo Arqueológico y Antropológico de Apurímac.

Adicionalmente, se eligieron 20 recursos turísticos para ser mostrados en la aplicación móvil como puntos de interés turístico de acuerdo a la geolocalización del turista, tales recursos turísticos se listan en la Tabla 1.

Por tanto, la muestra fue de n=25.

Tabla 1

Recursos considerados como puntos de interés turístico

Recursos turísticos	
Plaza principal de la ciudad de Abancay	Santuario Nacional Ampay
Parroquia Sagrario Catedral de Abancay	Aguas termales de Santo Tomás
Parque Micaela Bastidas	Conjunto arqueológico de Saywite
Calle Miscabamba	Aguas termales de Cconoc
Casa de David Samanez Ocampo	Mirador Capitán Rumi
Usno de Tamburco	El Cañón de Apurímac
Parque recreacional de Taraccasa	Laguna Rontoccocha
Hacienda Illanya	Túneles de Karkatera
Museo arqueológico y antropológico de Apurímac	Plaza de Tamburco
Puente colonial de Pachachaca	Capilla de Tamburco

3.4 Método de investigación

El tipo de investigación según el objeto de estudio es aplicada, teniendo en cuenta la naturaleza del problema que se concentra en estudiar y contribuir a la solución de un problema práctico (Vargas, 2009).

El nivel de investigación es explicativo ya que se busca dar un sentido de entendimiento a todo el estudio realizado y a su vez se maneja un proceso muy estructurado, (Hernández *et al.*, 2010).

El objetivo de este estudio es implementar el reconocimiento de objetos en un entorno de tiempo real de recursos turísticos, este reconocimiento podría ser hecho de diversas formas, sin embargo, se realizó utilizando *machine learning* para permitir el aprendizaje automático y que se pueda detectar el objeto aún en un estado difuso, como es, reconocerlo en su estado natural en medio de su propio ambiente; de esta forma, al reconocer el objeto se puede mostrar información detallada, así mismo, teniendo la ubicación del usuario, se logran desplegar puntos de interés a su alrededor, es decir, otros lugares turísticos o sitios de interés que se encuentran, de forma configurable, en un rango de kilómetros a

la redonda; por ello, al usar tecnologías diversas, es necesario que éstas tengan una arquitectura adecuada, que permita la comunicación entre sus módulos, los cuales finalmente se despliegan en una aplicación móvil para el usuario final.

3.5 Descripción detallada de métodos por objetivos específicos

Determinar el nivel de precisión del modelo entrenado en el reconocimiento de recursos turísticos de Apurímac al usar Machine Learning.

Para realizar el entrenamiento en *machine learning* y finalmente lograr determinar la precisión en el reconocimiento de imágenes, es necesario contar con una base de datos extensa que permita encontrar los patrones buscados y al compararla encontrar la solución. Existen diversos tipos de conjuntos de datos, cada uno de ellos diseñado o extraído a través del tiempo para cubrir una temática específica, como puede ser el caso de bases de datos de rostros, de letras y números, de objetos como autos, bicicletas, etc. Estas bases de datos son de acceso libre y otras de pago. Para esta investigación, dado que no se tiene el conjunto de datos de los atractivos turísticos elegidos, fue necesario construirla.

Se definieron cinco recursos turísticos dentro de la provincia de Abancay para ser reconocidos por el modelo de *machine learning* basado en red neuronal, estos recursos turísticos son: i) La piedra de Saywite, monolito ubicado en el complejo arqueológico de Saywite en el distrito de Curahuasi de la provincia de Abancay, ii) La estatua de Micaela Bastidas, ubicada en el Parque del mismo nombre en el distrito de Abancay, iii) La Pérgola de la Plaza de Armas de la ciudad de Abancay, iv) Momia enfardelada, ubicada en el Museo Arqueológico y Antropológico de Apurímac, localidad de Illanya del distrito de Abancay, y v) Nariguera y orejeras de oro, ubicadas en el Museo Arqueológico Antropológico de Apurímac, localidad de Illanya del distrito de Abancay.

Paso 1: Captura y colección de imágenes

Se realizó la captura de imágenes de cada recurso con el fin de obtener el objeto (recurso turístico) desde diversos ángulos, esta labor puede ser hecha tomando fotografías; en esta investigación se trabajó haciendo una filmación del recurso turístico desde diferentes ángulos, para posteriormente convertirlo a imágenes. Esto se llevó a cabo con la herramienta web en línea Online Convert, el resultado fueron las imágenes de los recursos turísticos en fotogramas.

Paso 2: Preprocesamiento de imágenes

Es una labor manual que constó de la verificación de cada una de las imágenes para depurar aquellas en las que existía una obstrucción del objeto a ser reconocido, como puede ser un árbol que en cierta toma cubre el objeto, una persona que transita y de igual forma obstruye la visión del objetivo. Al concluir esta labor se obtuvo un conjunto de datos de 2970 imágenes para todos los recursos turísticos trabajados. En el archivo train.txt se almacenaron todas las rutas relativas de cada imagen para el entrenamiento, como se ve en la *Figura 7*.

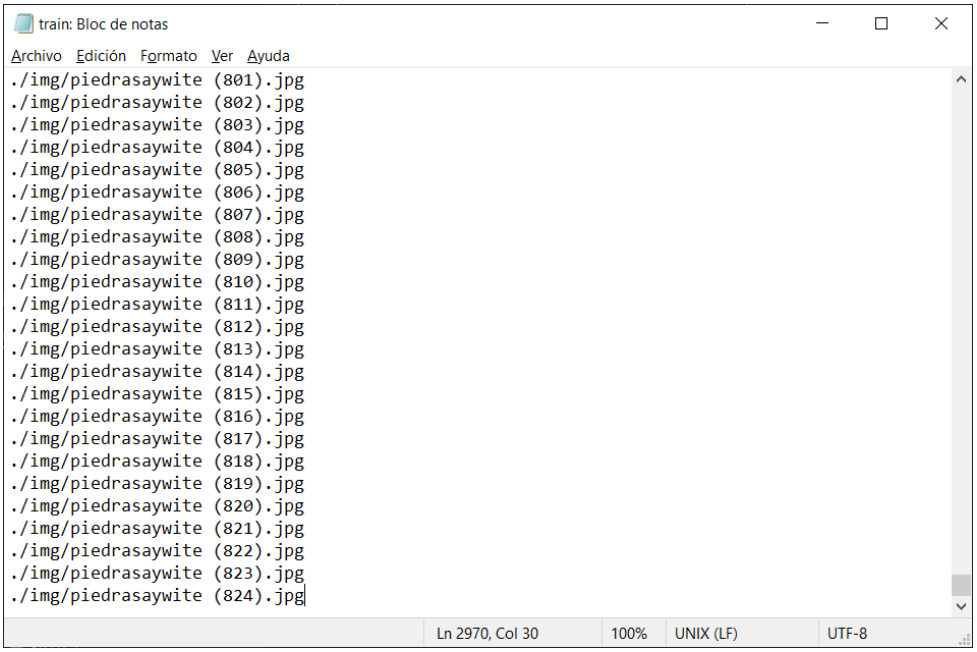


Figura 7. Muestra del archivo train.txt

Paso 3: Etiquetado de imágenes

Este proceso consiste en tomar el *dataset* creado y en cada imagen indicar la ubicación del objeto a reconocer (enmarcándolo) y asignarle la etiqueta (clase) respectiva, este proceso genera un archivo con las clases además de un archivo por cada imagen que usa YOLOv3 para el entrenamiento.

El etiquetado de imágenes es una labor que permite definir ciertos valores, figuras u objetos dentro de un conjunto de datos, en otras palabras, es nombrar un objeto para que luego se corresponda con lo predefinido en la base de datos. Esta labor también es manual y se relaciona con al archivo obj.names que contiene todas las etiquetas de imágenes a entrenar, que en este caso fueron cinco: “pergolaplazaabancay”, “piedrasaywite”, “estatuamicaela”, “momia” y “orejerasynarigueras”. Se realizó con el software LabelImg_v1.8.0.

Paso 4: Adaptación del archivo de configuración

Para lograr el reconocimiento de imágenes, se trabajó con el algoritmo de YOLOv3, el cual fue modificado, ajustándose a los parámetros propios del conjunto de imágenes a reconocer, puesto que por defecto viene con una base de datos precargada con la cual trabajar.

Se realizó la configuración del archivo yolov3.cfg, que contiene las configuraciones y parametrización para el entrenamiento. Los cambios centrales se refieren a atributos que se ajustan al modelo que se desea entrenar, la mayoría de atributos conserva la misma configuración original de YOLOv3, pero algunas fueron modificadas como el número máximo de lotes a ejecutar por iteración (max_batches) o el número de clases que reconocerá el modelo (classes), la captura de pantalla de este archivo se presenta en la *Figura 8*.

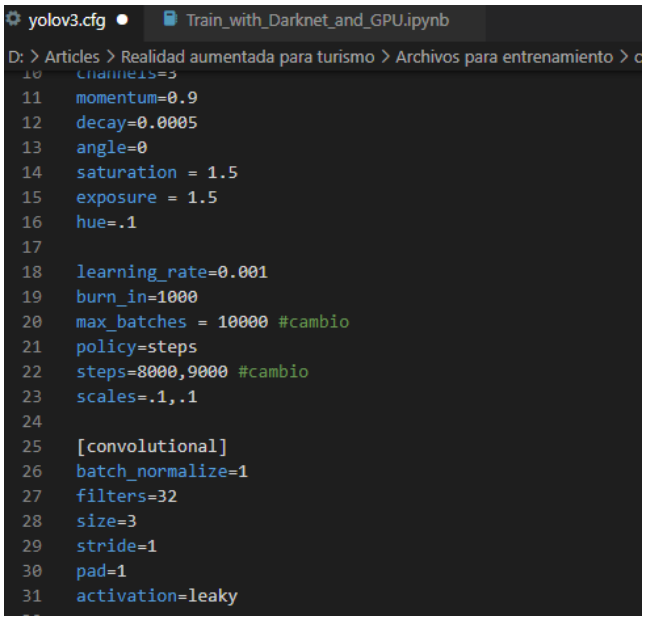


Figura 8. Archivo de configuración de YOLOv3 modificado

Paso 5: Entrenamiento en la Darknet.

El entrenamiento en la Darknet, se realiza con la configuración propuesta de YOLO, en su página oficial: https://pjreddie.com/darknet/yolo/

Figura 9. Página oficial de YOLOv3

Paso 6: Uso de GPU en Google Colaboratory y montado de Google Drive

– Se realizó la sincronización de archivos con la computadora local usando Google Drive (*Figura 10*).

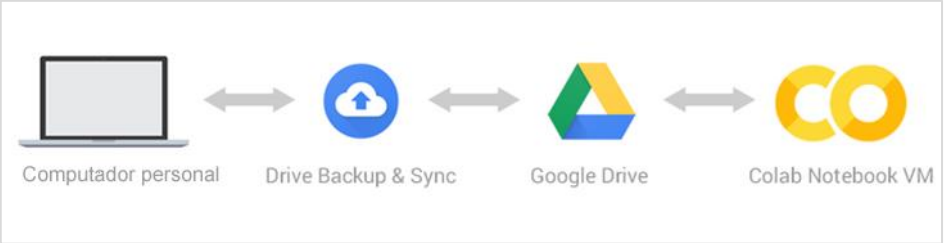


Figura 10. Configuración de copia de seguridad y sincronización con Colab

– Conectar Colab a Google Drive, en este paso es necesario mapear la carpeta de Google Drive donde se encuentren los archivos del entrenamiento, en este caso el archivo *obj.data* (*Figura 11*), que contiene el número de clases a entrenar (objetos) y rutas de archivos de configuración en Drive a los que apuntará Colab para el entrenamiento.

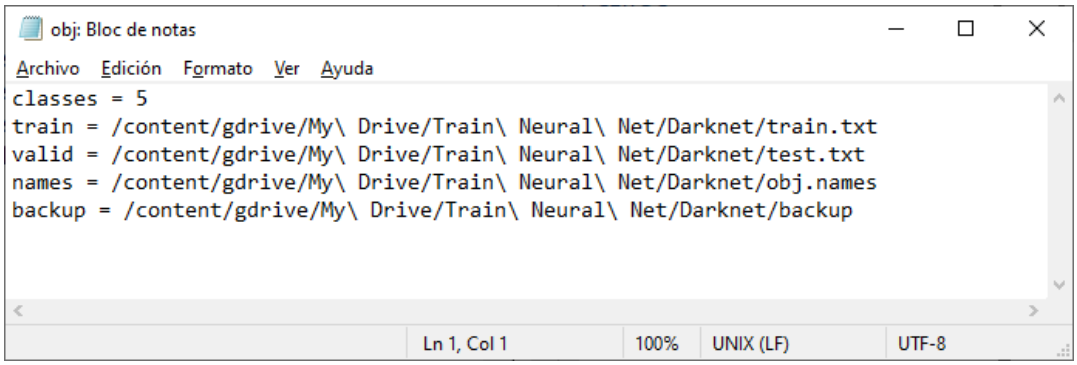


Figura 11. Archivo obj.data que define clases y rutas

Seguidamente se creó el cuaderno en Colab llamado Train_with_Darknet_and_GPU.ipynb (*Figura 12*) para acceder a Drive, hacer uso de GPU, instalar la Darknet y entrenar el modelo.

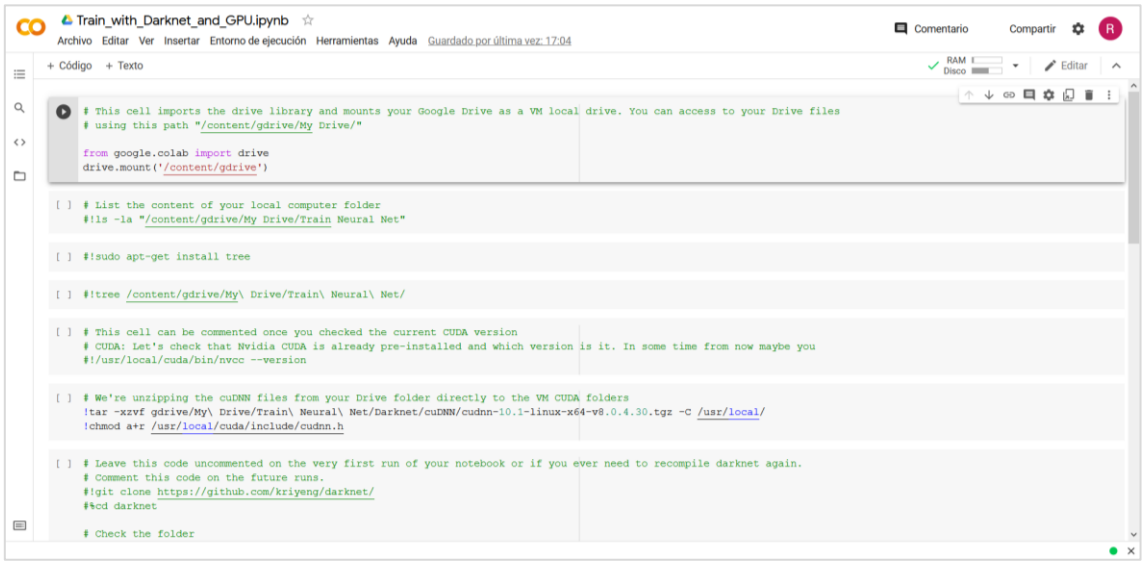


Figura 12. Configuración de Colab para desplegar Darknet

Por último, con el modelo entrenado se procedió a calcular el nivel de precisión.

Determinar en tiempo real, los puntos de interés turístico cercanos, en base a la geolocalización y orientación de la cámara del móvil, a través de una app móvil con realidad aumentada

Para realizar esta tarea se deben leer datos de los sensores del móvil como su geolocalización y su orientación. En base a estos datos y realizando diversos cálculos se muestran en una aplicación móvil con realidad aumentada los puntos de interés turísticos cercanos y ordenados de acuerdo a su posición respecto al rumbo del dispositivo móvil, para que el usuario pueda desplegar su información.

En esta labor se realizaron dos tareas fundamentales:

Paso 1: Determinar la distancia de dos puntos sobre una esfera

Uno de los métodos más utilizados para este propósito es la fórmula de Haversine que calcula la distancia geográfica en la tierra y cuya fórmula se presenta en la *Figura 13*.

$$d = 2r\ arc\sin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 \times \cos\varphi_2 \times sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Figura 13. Fórmula de Haversine

Donde:

r: radio de la Tierra. El radio medio de la Tierra es 6371 km.

φ1, φ2: son la latitud del punto 1 y la latitud del punto 2 (en radianes).

λ1, λ2: son la longitud del punto 1 y la longitud del punto 2 (en radianes).

Paso 2: Hallar la dirección de visualización de un punto respecto al enfoque de la cámara

En este caso se ha considerado el norte magnético, como base para desarrollar el algoritmo, donde primero se calculó el ángulo del foco de la cámara, respecto al polo ya mencionado, luego se calculó el ángulo de los puntos de interés registrados en el sistema a "x" kilómetros a la redonda, respecto al hemisferio norte y a la posición del dispositivo móvil, en base al GPS y, finalmente se calculó el ángulo diferencial entre el foco de la cámara y los puntos de interés; de acuerdo a eso, se hizo un cálculo adicional para determinar si cada punto de interés cercano se encontraba al centro, derecha o izquierda de la orientación del móvil.

Definir una arquitectura que permita la interacción entre Machine Learning, Realidad Aumentada, Aplicación Web y Aplicación Móvil en el reconocimiento de recursos turísticos de Apurímac

La aplicación completa involucra varias tecnologías como *machine learning*, realidad aumentada, apps móviles y aplicaciones web. Por lo que se desarrollaron diversos componentes los cuales se comunican entre sí para lograr mostrar al usuario información de interés turístico para recursos reconocidos por la aplicación o recursos detectados de acuerdo a la proximidad y orientación de la cámara del dispositivo móvil.

Paso 1: Definir los componentes

La aplicación completa se divide en los siguientes componentes:

El primer componente es el encargado de realizar el reconocimiento de objetos, en este caso recursos turísticos, para lo cual se utilizó técnicas de *machine learning*, lo cual implica el entrenamiento con grandes volúmenes de datos de la red neuronal y una vez terminado el entrenamiento se realiza la etapa de pruebas, dejando el modelo listo para hacer el reconocimiento de los recursos turísticos para los cuales fue entrenado. Todo esto se realizó en el lenguaje de programación Python, por ser uno de los más utilizados para *machine learning*.

El segundo componente es el responsable de gestionar el contenido que se desplegará en la aplicación móvil y que se corresponde con cada recurso turístico, como el nombre, descripción, coordenadas e información turística relacionada. Este componente está disponible para el administrador de la aplicación por lo que se desarrolló en entorno web, en el lenguaje de programación PHP.

El tercer componente del sistema es el destinado al usuario final, con técnicas de realidad aumentada, que le permitirá visualizar información turística en el móvil apuntando con la cámara un recurso turístico o a través de geolocalización, mostrar puntos de interés turístico, de acuerdo a su cercanía y orientación del móvil. Para esto se desarrolló una aplicación móvil híbrida con Ionic, el cual permite construir aplicaciones móviles utilizando HTML, CSS y JavaScript.

Paso 2: Definir los patrones de arquitectura

Los patrones arquitectónicos, los cuales son similares a los patrones de diseño de software, pero con un alcance más amplio; por cada componente se describen de la siguiente manera:

El primer componente de reconocimiento de objetos fue principalmente desarrollado bajo programación estructurada.

El segundo componente de administración de contenidos se desarrolló bajo el patrón Modelo Vista Controlador (MVC), el cual ya viene integrado en el *framework* Laravel.

El tercer componente que es la aplicación móvil fue desarrollado con el patrón Modelo Vista Modelo de Vista (MVVM), ya que es un estándar para la implementación de aplicaciones para Android.

En conjunto la disposición de los componentes del sistema se asemeja a una Arquitectura Orientada a Servicios (SOA).

Paso 3: Definir la comunicación entre componentes

La comunicación de los componentes antes descritos se realizó mediante la arquitectura REST (REpresentational State Transfer) la cual utiliza para la comunicación el protocolo HTTP y sirve los datos mediante el formato JSON.

Por lo que se implementaron API de RESTful que cumplen con la arquitectura REST, para comunicar los tres componentes del sistema.

Con la tecnología mencionada, se expusieron servicios en el componente de reconocimiento de objetos por medio de *machine learning*, para ser consumidos por la aplicación de usuario final.

Otro componente en el que se expusieron servicios a través de un API de RESful fue el de administración de contenidos, el cual provee a la aplicación de usuario final la información de los puntos de interés.

Finalmente, el componente de la app móvil consume los recursos de las dos APIs anteriores. Esto le permite reconocer recursos turísticos enfocados por la cámara del móvil y una vez reconocidos muestra información relacionada.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1 Determinar el nivel de precisión del modelo entrenado en el reconocimiento de recursos turísticos de Apurímac al usar Machine Learning

Para este objetivo, se definieron cinco recursos turísticos de la provincia de Abancay para el entrenamiento y reconocimiento del modelo, los cuales son: La piedra de Saywite, la Momia enfardelada del Museo Arqueológico y Antropológico, la Nariguera y Orejeras del Museo Arqueológico y Antropológico, la Pérgola de la Plaza de Armas de Abancay y la estatua de Micaela Bastidas del parque del mismo nombre.

Debido a que no se cuenta con un *dataset* de tales recursos turísticos, se procedió a elaborarlo. Para esto se realizó la filmación de cada punto mencionado anteriormente desde diversos ángulos para posteriormente convertirlos a fotogramas con la herramienta en línea Online Convert. De este proceso y luego de depurar las imágenes, se obtuvieron en total, para los cinco recursos turísticos, 2970 imágenes divididas en 1400 de la estatua de Micaela Bastidas, 170 de la Momia enfardelada, 141 de la Nariguera y Orejeras, 435 de la Pérgola de la Plaza de Armas y 824 de la piedra de Saywite.

El proceso de etiquetado de imágenes indica en cada imagen del *dataset* la ubicación de los recursos turísticos y asigna la clase correspondiente. Las clases o etiquetas que se utilizaron para designar cada recurso son: “pergolaplazaabancay”, “piedrasaywite”, “estatuamicaela”, “momia” y “orejerasynarigueras”.

Para etiquetar las imágenes se utilizó el programa LabelImg, el cual crea las etiquetas en un archivo llamado classes.txt, el cual se cambió de nombre a obj.names. Además, LabelImg genera un archivo .txt por cada imagen procesada, que contiene la etiqueta y las

coordenadas del objeto dentro de la imagen. Todo lo anterior lo hace en el formato utilizado por YOLO. En la *Figura 14* se puede ver el proceso de etiquetado.

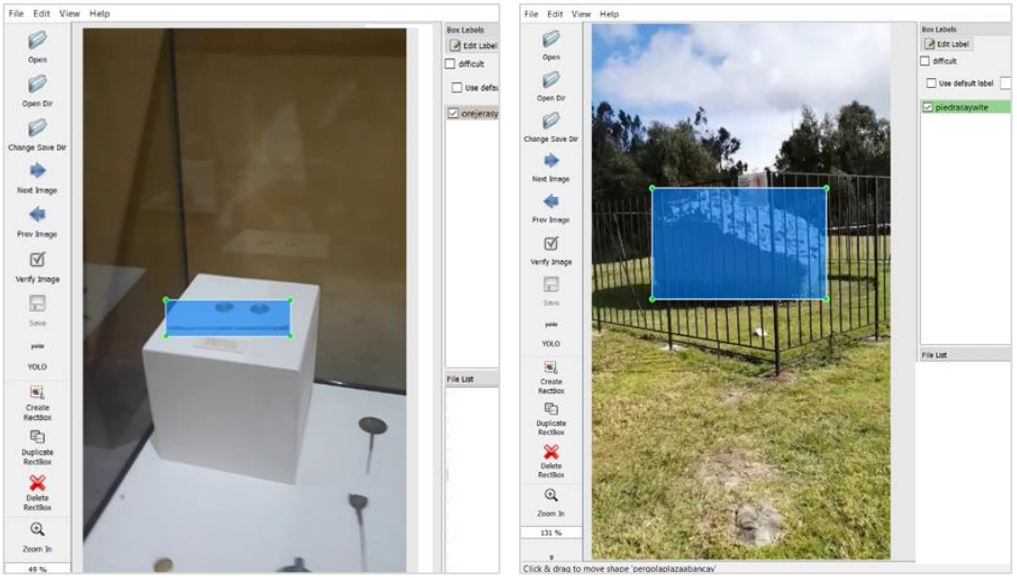


Figura 14. Etiquetado de recursos turísticos

Con el conjunto de datos etiquetado, se procede al entrenamiento de la red neuronal, para lo cual se utilizó el algoritmo de YOLOv3 con la implementación del *framework* de redes neuronales Darknet.

Se instaló Darknet en una máquina virtual de Google Colab para aprovechar la GPU que nos ofrece y acelerar el tiempo de entrenamiento. Además, se vinculó Google Colab con Google Drive para allí organizar nuestras carpetas y archivos necesarios para el entrenamiento, como se muestra en la Figura 15.

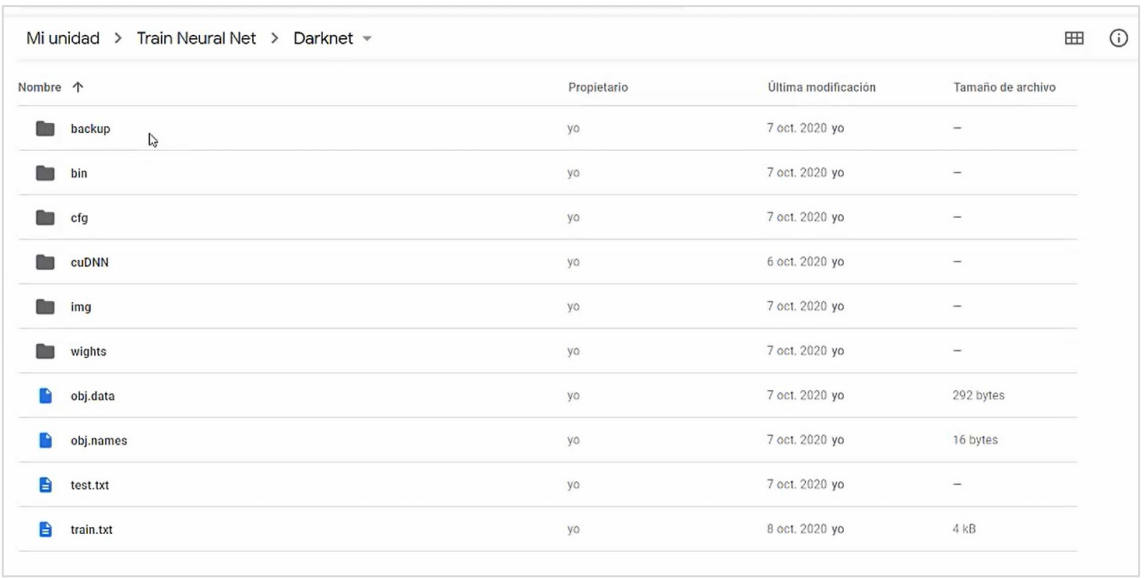


Figura 15. Estructura de carpetas para el entrenamiento en Google Drive

La estructura de carpetas y archivos mostrada es necesaria para que Darknet realice el entrenamiento de una red neuronal para el reconocimiento de nuestros recursos turísticos, a continuación, se explica cada elemento:

- Archivo train.txt: indica la ruta a cada una de las imágenes del *dataset* con las que el modelo se va a entrenar, son rutas relativas a la raíz de la máquina virtual de Colab.
- Archivo test.txt: indica la ruta a cada una de las imágenes con las que el modelo se va a validar, son rutas relativas a la raíz de la máquina virtual de Colab.
- Archivo obj.names: contiene los nombres de la clases a las que pertenecen cada uno de los recursos turísticos, en este caso: pergolaplazaabancay, piedrasaywite, estatua-micaela, momia y orejerasynarigueras. Este archivo se generó en el etiquetado como classes.txt, el cual se renombró como obj.names.
- Archivo obj.data: indica la cantidad de clases, en nuestro caso 5, así como las rutas relativas a los archivos de entrenamiento: train.txt, test.txt, obj.names, así como a la carpeta backup.
- Carpeta backup: se crea y se deja vacía, en esta carpeta se almacenarán los pesos de los modelos entrenados. Cada 100 iteraciones se sobrescribe un archivo last y cada 1000 iteraciones se crea un archivo, y cuando finaliza el entrenamiento se crea el archivo final.
- Carpeta weights: Esta carpeta contiene un modelo con pesos preentrenados para las capas convolucionales, el cual sirve para iniciar con el entrenamiento. Para el caso de YOLOv3 el archivo se llama darknet53.conv.74.
- Carpeta img: contiene todas las imágenes para el entrenamiento con sus correspondientes archivos de etiquetado.
- Carpeta cuDNN: contiene el archivo controlador de dispositivo gráfico de la máquina virtual de Colab, el cual se descarga desde la página de Nvidia y nos permitirá utilizar la GPU en Colab.
- Carpeta bin: contiene la Darknet compilada, para no tener que instalarla en Colab cada vez que se continúe con el entrenamiento.
- Carpeta cfg: contiene el archivo yolov3.cfg, el cual es el archivo de configuración para el algoritmo YOLOv3.

El archivo de configuración yolov3.cfg se debe personalizar a nuestras características específicas, de acuerdo a las sugerencias en la documentación se realizaron cambios en los

parámetros *batch, subdivisions, max_batches, steps, width, height, classes, filters*, con los datos propios de nuestro entrenamiento.

Las instrucciones para el entrenamiento y validación se realizaron mediante un *notebook* de Colab, ya que se utilizó una máquina virtual de la nube de Google. En este *notebook* primero se configuró para que utilice GPU. Luego se monta Google Drive para que sea accesible como unidad desde Google Colab, esto necesita de un código de autorización. Ahora se verifica la versión de CUDA que tiene la máquina virtual de Colab para poder descargar su controlador adecuado, la versión fue 10.1.243 por lo que se descargó el archivo cudnn-10.1-linux-x64-v8.0.4.30.tgz y se colocó en la carpeta cuDNN en Drive, para posteriormente descomprimirlo en la carpeta local de la máquina virtual de Colab. Posteriormente se clonó un *fork* de Darknet desde su repositorio en GitHub, se hizo un cambio de rama a la versión que ya está preparada para trabajar con Google Colab y se instaló; una vez instalada se copió a nuestra carpeta bin de Google Drive para no tener que instalarla cada vez que empecemos el entrenamiento. La instalación de Darknet sólo se realizó la primera vez, debido a que se guardó en Google Drive el compilado, para los siguientes entrenamientos sólo se copiará este compilado de Google Drive al sistema de archivos de la máquina virtual. Para un tiempo de ejecución más corto, se copiaron las imágenes de entrenamiento de Google Drive a la máquina virtual de Google Colab en una carpeta img.

Con todo lo anterior se dio inicio al entrenamiento, con el comando detector train que necesita tres parámetros: el primero es la ruta al archivo obj.data del Drive, el segundo es la ruta al archivo de configuración yolov3.cfg del Drive y el tercero dependerá de si estamos entrenando por primera vez o no; si es la primera vez será la ruta al archivo preentrenado darknet53.conv.74 que se encuentra en la carpeta weights del Drive (Figura 16); pero si no es la primera vez, porque se debe continuar el entrenamiento desde un punto anterior, se pasa la ruta al archivo yolov3_last.weights de la carpeta backup de Drive (Figura 17).

```
!./darknet detector train "/content/gdrive/My Drive/Train Neu-
ral Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neu-
ral Net/Darknet/cfg/yolov3.cfg" "/con-
tent/gdrive/My Drive/Train Neural Net/Darknet/weights/dark-
net53.conv.74" -dont_show -map
```

Figura 16. Comando para entrenamiento por primera vez

```
!./darknet detector train "/content/gdrive/My Drive/Train Neu-
ral Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neu-
ral Net/Darknet/cfg/yolov3.cfg" "/con-
tent/gdrive/My Drive/Train Neural Net/Dark-
net/backup/yolov3_last.weights" -dont_show -map
```

Figura 17. Comando para continuar con el entrenamiento desde un punto anterior

El proceso de entrenamiento para los cinco recursos turísticos (clases) realizó 9000 iteraciones en total, lo cual duró 4 días, con un promedio de 12 horas por día, haciendo un total aproximado de 50 horas.

Al final del entrenamiento se generó el archivo yolov3_final.weights, dentro de la carpeta *backup* de Drive, definido así en obj.data, este archivo contiene los pesos para las capas convolucionales de la red neuronal entrenada de YOLOv3.

Este *notebook*, al que se nombró como Train_with_Darknet_and_GPU.ipynb, se muestra de manera detallada y completa en el Anexo 1.

Para conocer el nivel de precisión del modelo entrenado, se ejecuta el comando detector map de Darknet, pasándole como parámetros la ruta archivo obj.data, la ruta al archivo yolov3.cfg y como tercer parámetro la ruta al archivo yolov3_final.weights, que contiene los pesos finales del entrenamiento, como se observa en la Figura 18.

```
!./darknet detector map "/content/gdrive/My Drive/Train Neu-
ral Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neu-
ral Net/Darknet/cfg/yolov3.cfg" "/con-
tent/gdrive/My Drive/Train Neural Net/Darknet/backup/yolov3_fi-
nal.weights"
```

Figura 18. Comando para validar el modelo

En la Figura 19 se observa el resultado de la validación del modelo, mediante el promedio de precisión mAP (mean average precision), el cual representa el valor medio de las precisiones promedio para cada clase, donde la precisión promedio es el valor medio de 11 puntos en la curva PR (Precisión-Recall) para cada posible umbral (cada probabilidad de detección) para la misma clase.

```
calculation mAP (mean average precision)...
2972
 detections_count = 3243, unique_truth_count = 2970
class_id = 0, name = pergolaplazaabancay, ap = 100.00%       (TP = 435, FP = 0)
class_id = 1, name = piedrasaywite, ap = 100.00%      (TP = 824, FP = 0)
class_id = 2, name = estatuamicaela, ap = 99.93%      (TP = 1400, FP = 1)
class_id = 3, name = momia, ap = 100.00%       (TP = 164, FP = 0)
class_id = 4, name = orejerasynarigueras, ap = 100.00%      (TP = 141, FP = 0)

 for thresh = 0.25, precision = 1.00, recall = 1.00, F1-score = 1.00
 for thresh = 0.25, TP = 2964, FP = 1, FN = 6, average IoU = 87.85 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.999850, or 99.99 %
Total Detection Time: 149.000000 Seconds
```

Figura 19. Resultados de la precisión obtenida en el entrenamiento en Darknet

Contrastación de hipótesis

Para contrastar la hipótesis se verificó el nivel de precisión del modelo entrenado de tres maneras distintas: i) utilizando la validación de Darknet mediante detector map, la cual como se observó anteriormente, calcula el mAP o promedio de precisión de un conjunto de imágenes de validación; ii) además se utilizó el comando de Darknet detector test, el cual evalúa al modelo pasándole una imagen; iii) finalmente se evaluó el modelo con pruebas reales de detección, en la que se utilizan las imágenes captadas por la cámara de un móvil.

Para lo cual se realizaron las mediciones, las cuales se muestran en la Tabla 2, donde la primera medición es de la validación del modelo con detector map, las siguientes 30 mediciones corresponden a la validación con detector test pasándole como parámetro 6 imágenes por cada clase y las últimas 20 mediciones son pruebas reales *in situ*, 4 veces para cada clase, apuntando con la cámara del móvil hacia el recurso turístico.

Tabla 2

Mediciones de la precisión del modelo

Medición	Precisión	Medición	Precisión	Medición	Precisión	Medición	Precisión
1	99	14	95	27	100	40	100
2	96	15	100	28	100	41	100
3	95	16	69	29	100	42	100
4	68	17	98	30	100	43	100
5	88	18	100	31	99	44	100
6	75	19	100	32	100	45	100
7	100	20	82	33	100	46	100
8	83	21	67	34	100	47	100
9	100	22	93	35	100	48	100
10	59	23	85	36	100	49	100
11	81	24	97	37	100	50	100
12	99	25	96	38	100	51	100
13	93	26	66	39	100		

Hipótesis

Al aplicar *machine learning*, se logra un nivel de precisión adecuado del modelo entrenado para el reconocimiento de recursos turísticos de Apurímac.

1. Hipótesis estadísticas

Hipótesis nula:

$H_0 : \mu \leq 90$ La media de las precisiones del modelo entrenado para el reconocimiento de recursos turísticos es igual o inferior a 90%

Hipótesis alterna:

$H_1 : \mu > 90$ La media de las precisiones del modelo entrenado para el reconocimiento de recursos turísticos es superior a 90%

2. Nivel de significancia

Se establece el nivel de significancia en 5%. α=5% ó α=0.05

3. Estadístico de prueba

Se establece para este caso la prueba de hipótesis para medias en muestras pequeñas desconociendo la desviación estándar de la población, para lo cual utilizamos la prueba T-

Student para una muestra. El cálculo del estadístico lo hacemos en el programa estadístico Minitab 18 y los resultados se muestran en la Tabla 3.

Tabla 3

Resultados de la prueba de hipótesis

Hipótesis nula H_0: $\mu \leq 90$ **Hipótesis alterna** H_1: $\mu > 90$	
Valor T	Valor p
2.45	0.009

4. Lectura del p-valor

De acuerdo a la tabla anterior observamos que el $p - valor = 0.009$

5. Decisión estadística

Dado que $p - valor < \alpha$, entonces no se puede afirmar H_0 (hipótesis nula) y se acepta H_1 (hipótesis alterna).

Existe evidencia de que la media de las precisiones del modelo entrenado para el reconocimiento de recursos turísticos en Apurímac, es mayor a 90%, con un nivel de significancia de 5%.

Por lo que se comprueba la hipótesis que indica que al aplicar *machine learning*, se logra un nivel de precisión adecuado del modelo entrenado para el reconocimiento de recursos turísticos de Apurímac.

4.2 Determinar en tiempo real, los puntos de interés turístico cercanos, en base a la geolocalización y orientación de la cámara del móvil, a través de una app móvil con realidad aumentada

Una de las funciones de la aplicación móvil es ubicar al usuario geográficamente y en función a esta posición, mostrar los puntos de interés turístico que puedan estar a su alrededor en un radio determinado, para ello es necesario calcular la distancia entre dos puntos. Esta labor sobre un plano es bastante más sencilla, por ejemplo, utilizando coordenadas UTM simplemente se aplica el teorema de Pitágoras, sin embargo, cuando estos dos puntos se ubican sobre la esfera terrestre, el cálculo de distancias en una esfera debe considerar la geometría esférica, el estudio de formas en la superficie de una esfera. En este caso la posición del usuario está dada en coordenadas geográficas (latitud y longitud) por el sensor del equipo móvil, así como también las posiciones de los puntos de interés

turístico dentro del sistema de gestión se ingresaron a través de Google Maps en coordenadas geográficas.

Sumado a ello, al usar la cámara del dispositivo móvil, es necesario aplicar la ubicación de un punto de interés alrededor a "x" kilómetros a la redonda, teniendo en cuenta la orientación y el ángulo en que la cámara debe posicionarse para lograr visualizar estos puntos a su alrededor, es decir, principalmente validar el ángulo del foco de la cámara.

Para lograr el objetivo, primero se obtiene la geolocalización en tiempo real del dispositivo móvil, esto se hace a través del sensor de geolocalización del cual se obtienen la latitud y longitud. Además, mediante el sensor geomagnético o magnetómetro, que cumple la función de brújula en el móvil, se obtiene la orientación del móvil con respecto al norte magnético.

Con estos datos, se calculan las distancias desde la posición actual del móvil con cada punto de interés. Esta tarea en realidad es determinar la distancia entre dos puntos sobre una esfera. La geometría esférica considera la trigonometría esférica que se ocupa de las relaciones entre funciones trigonométricas para calcular los lados y ángulos de polígonos esféricos. Estos polígonos esféricos están definidos por una serie de grandes círculos que se cruzan en una esfera.

El método más común para calcular la distancia geográfica en la tierra es la fórmula de Haversine. Si se tiene dos valores diferentes de latitud y longitud de dos puntos diferentes en la tierra, entonces, con la ayuda de la fórmula de Haversine, se puede calcular fácilmente la distancia del círculo máximo (la distancia más corta entre dos puntos en la superficie de una esfera). El término Haversine fue acuñado por el profesor James Inman en 1835, y es usada frecuentemente para desarrollar aplicaciones GIS (Sistema de Información Geográfica) o analizar rutas y campos. Si bien la fórmula más conocida es la mostrada en la Figura 13, en esta investigación se utilizó la siguiente variante:

$$d = 2r\ arc\sin\sqrt{\frac{1}{2} - \frac{\cos(\varphi_1 - \varphi_2)}{2} + \cos\varphi_2 \times \cos\varphi_1 \times \frac{(1 - \cos(\lambda_1 - \lambda_2))}{2}}$$

Figura 20. Fórmula de Haversine utilizada

Donde:

r: radio de la Tierra. El radio medio de la Tierra es 6371 km.

φ1, φ2: son la latitud del punto 1 y la latitud del punto 2 (en radianes).

λ1, λ2: son la longitud del punto 1 y la longitud del punto 2 (en radianes).

Una vez calculadas las distancias a cada punto de interés turístico, se almacenaron para ser mostrados sólo los puntos de interés que están en un radio menor a "x" km a la redonda.

El rumbo, que es el ángulo de la orientación medido en sentido horario desde el norte magnético hacia la línea formada por el móvil con el punto de interés, se calculó de acuerdo a la trigonometría esférica, con la siguiente fórmula:

$$x = \cos\varphi_1 \times \sin\varphi_2 - \sin\varphi_1 \times \cos\varphi_2 \times \cos(\lambda_2 - \lambda_1)$$
$$y = \sin(\lambda_2 - \lambda_1) \times \cos\varphi_2$$
$$\theta = \operatorname{atan}2(y, x)$$

Figura 21. Fórmula para el cálculo de la orientación inicial o rumbo

Donde:

θ: es el rumbo u orientación inicial expresado en radianes.

φ1, λ1: son la latitud y longitud de la ubicación del móvil (en radianes).

φ2, λ2: son la latitud y longitud del punto de interés (en radianes).

El rumbo θ obtenido se convirtió a grados centígrados y se normalizó el resultado a un rumbo de brújula en el rango de 0° a 360°. Se tiene entonces el ángulo formado por el norte con la orientación del móvil y recientemente los ángulos formados por el norte y las líneas del móvil y los puntos de interés. Entonces se calculó cada diferencia entre estos ángulos para determinar la posición de cada punto de interés respecto de la orientación del móvil. Esta diferencia permitió mostrar los puntos de interés en el centro de la pantalla si se alineaba con la orientación del móvil o mostrar indicadores que se encuentran a la derecha o a la izquierda con respecto a la orientación del móvil en tiempo real.

Figura 22. Indicadores de posición de puntos de interés

El procedimiento descrito se implementó en una app móvil híbrida multiplataforma que utiliza HTML5, CSS, JavaScript y Cordova como base, realizada en Ionic con Angular, la cual de acuerdo a la localización del móvil determinada por el sensor GPS y la orientación dada por el magnetómetro, muestra mediante realidad aumentada indicadores de los puntos de interés que se encuentran en un radio entre 1 hasta 9 km establecidos por el usuario, estos puntos se muestran con indicadores para precisar si están a la derecha, izquierda o centro, de acuerdo a la orientación del móvil (Figura 22). La información de cada punto de interés turístico fue cargada previamente a través del sistema web de administración de recursos turísticos (Figura 23).

Lista de puntos de interes

Ingrese datos de búsqueda (Enter)

Nombre	Descripción	Dirección	Categoría	Latitud	Longitud
Casa de David Samanez Ocampo	Declarada monumento histórico mediante Ley N° 24677, publicado en el diario oficial «El Peruano» del 31 de Mayo de 1987, por haber sido la vivienda de David Samanez Ocampo, Presidente de la Junta de Gobierno de 1931. Ubicado en la esquina formada por el Jr. Arequipa y la Av. Núñez en la capital del departamento de Apurímac, es un homenaje a la memoria de tan notable protagonista de la historia peruana.	Intersección Jr. Arequipa con Av. Núñez	General	-13.636251260275134	-72.87698108949661
Calle Miscabamba	Las más antigua de Abancay, si logras llegar hacia el Parque Micaela Bastidas, entonces no hay excusa para no recorrer este atractivo turístico de Abancay. Hacia el lado izquierdo del parque comienza la calle Miscabamba, cada vez hay construcciones nuevas, solo quedan algunas casonas antiguas con sus típicas puertas de madera y su techo de teja.	Calle Miscabamba	General	-13.635856359592067	-72.87512768306732
Parque Micaela Bastidas	Situado en la ciudad de Abancay a cuatro cuadras de la plaza principal, el parque fue construido en honor a una distinguida heroína, que nació en esta ciudad (Micaela Bastidas) puedes llegar caminando, solo toma unos 15 minutos desde la plaza principal.	Av. Arenas S/N	General	-13.635885032271158	-72.87476558485031
Parroquia Sagrario Catedral de Abancay	La primera piedra de la Iglesia Catedral, fue colocada en 1645 por el P. Fr. Domingo Cabrera de Lartaún, se caracteriza por una arquitectura simple, tiene una sola torre con un campanario. La patrona Titular de todo el Valle de Abancay es la Virgen del Rosario que se celebra el 07 de octubre.	Plaza de Armas - Jr. Lima s/n Abancay - Apurímac.	General	-13.637227431077337	-72.87841070690155
Plaza de Armas de la ciudad de Abancay	Un espacio pequeño en comparación con otras plazas, pero muy acogedor y a pocos metros se ubica la Catedral de la Virgen del Rosario que data de 1645.	Jr. Lima entre la Av. Mariño y el Jr. Cusco	General	-13.637308235422198	-72.87886131801605

Figura 23. Información de puntos de interés turístico

El código completo de la implementación de lo mencionado anteriormente, se muestra en el Anexo 3.

Contrastación de hipótesis

Para contrastar esta hipótesis se realizaron pruebas con la *app* móvil, en radio de 1 km, en diferentes zonas de la ciudad, se anotaron los puntos de interés turístico identificados y se contrastaron con los datos de Google Maps para determinar si fue un acierto o no.

Luego del proceso anterior se anotaron los datos como se muestran en la Tabla 4, donde el número 1 representa acierto y el 0 representa no acierto.

Tabla 4

Observaciones de aciertos en la identificación de puntos de interés turístico

Observación	Acierto	Observación	Acierto	Observación	Acierto
1	1	11	1	21	1
2	1	12	1	22	1
3	1	13	1	23	1
4	1	14	1	24	1
5	1	15	1	25	1
6	1	16	1	26	1
7	1	17	1	27	1
8	1	18	1	28	1
9	1	19	1	29	1
10	1	20	1	30	1

Hipótesis

Al aplicar técnicas de geoposicionamiento, la *app* móvil identifica correctamente los puntos de interés turístico cercanos en tiempo real.

1. Hipótesis estadísticas

Hipótesis nula:

$H_0 : \rho \leq 0.9$ La proporción de aciertos de la *app* móvil identificando correctamente los puntos de interés turístico cercanos en tiempo real es igual o inferior a 0.9

Hipótesis alterna:

$H_1 : \rho > 0.9$ La proporción de aciertos de la *app* móvil identificando correctamente los puntos de interés turístico cercanos en tiempo real es superior a 0.9

2. Nivel de significancia

Se establece el nivel de significancia en 5%. α=5% ó α=0.05

3. Estadístico de prueba

Se establece para este caso la prueba de hipótesis para la proporción de un evento observado en una sola muestra, para lo cual utilizamos la prueba de 1 proporción. El cálculo del estadístico lo hacemos en el programa estadístico Minitab 18 y los resultados se muestran en la Tabla 5.

Tabla 5

Resultados de la prueba de hipótesis

Hipótesis nula **Hipótesis alterna**	**H_0: p ≤ 0.9** **H_1: p > 0.9**
Valor p	0.042

4. Lectura del p-valor

De acuerdo a la tabla anterior observamos que el $p - valor = 0.042$

5. Decisión estadística

Dado que $p - valor < \alpha$, entonces no se puede afirmar H_0 (hipótesis nula) y se acepta H_1 (hipótesis alterna).

Existe evidencia de que la proporción de acierto de la *app* móvil identificando correctamente los puntos de interés turístico cercanos en tiempo real es mayor a 0.9 (90%), con un nivel de significancia de 5%.

Por lo que se comprueba la hipótesis que indica que, al aplicar técnicas de geoposicionamiento, la *app* móvil identifica correctamente los puntos de interés turístico cercanos en tiempo real.

4.3 Definir una arquitectura que permita la interacción entre Machine Learning, Realidad Aumentada, Aplicación Web y Aplicación Móvil en el reconocimiento de recursos turísticos de Apurímac

En la Figura 24, se muestra la arquitectura de la aplicación, compuesta por tres sistemas centrales: i) Sistema de reconocimiento de recursos turísticos, ii) Sistema de gestión de contenido turístico y iii) Aplicación móvil de cliente final con realidad aumentada.

El primer sistema es el de reconocimiento de los objetos, en este caso son objetos reales de recursos turísticos. Primero se realizó el entrenamiento, mediante *machine learning*, del modelo de red neuronal convolucional, el cual se implementó aplicando el algoritmo YOLOv3 para detección de imágenes, usando TensorFlow por debajo. El entrenamiento del modelo creado se trabajó con el *framework* Darknet, haciendo uso de GPU sobre Google Colab, todo bajo el lenguaje Python. A continuación, se introdujo el modelo validado en una RESTful API, la cual recibe imágenes enviadas por la aplicación móvil y responde si ha reconocido algún recurso turístico con su correspondiente información turística; esta API fue construida en Python con el *framework* Flask.

El segundo sistema es el sistema de gestión de contenido, el cual gestiona la información turística de los recursos, que será mostrada mediante realidad aumentada en la *app* móvil; además de la información multimedia para cada recurso turístico, se utilizó el API de Google Maps para establecer su posición geográfica, este sistema de gestión fue hecho en lenguaje PHP con el *framework* Laravel y el almacenamiento en MySQL.

Finalmente, el tercer sistema es una aplicación móvil diseñada para el usuario final, en el cual se implementó la *app* que consume los datos del sistema de reconocimiento de objetos y del sistema de gestión de contenido turístico, para ser mostrados mediante realidad aumentada en la pantalla del móvil. Esta *app* toma las imágenes de la cámara del móvil y las envía a la API de reconocimiento, para que, de acuerdo al recurso, se muestre su información turística; además de acuerdo a los datos del GPS y del magnetómetro identifica los puntos de interés turístico en un radio de kilómetros de su ubicación, pudiendo ver la información mediante realidad aumentada. Esta *app* se implementó con Ionic que se usa para el desarrollo de aplicaciones móviles híbridas, Angular y la *app* se generó para Android.

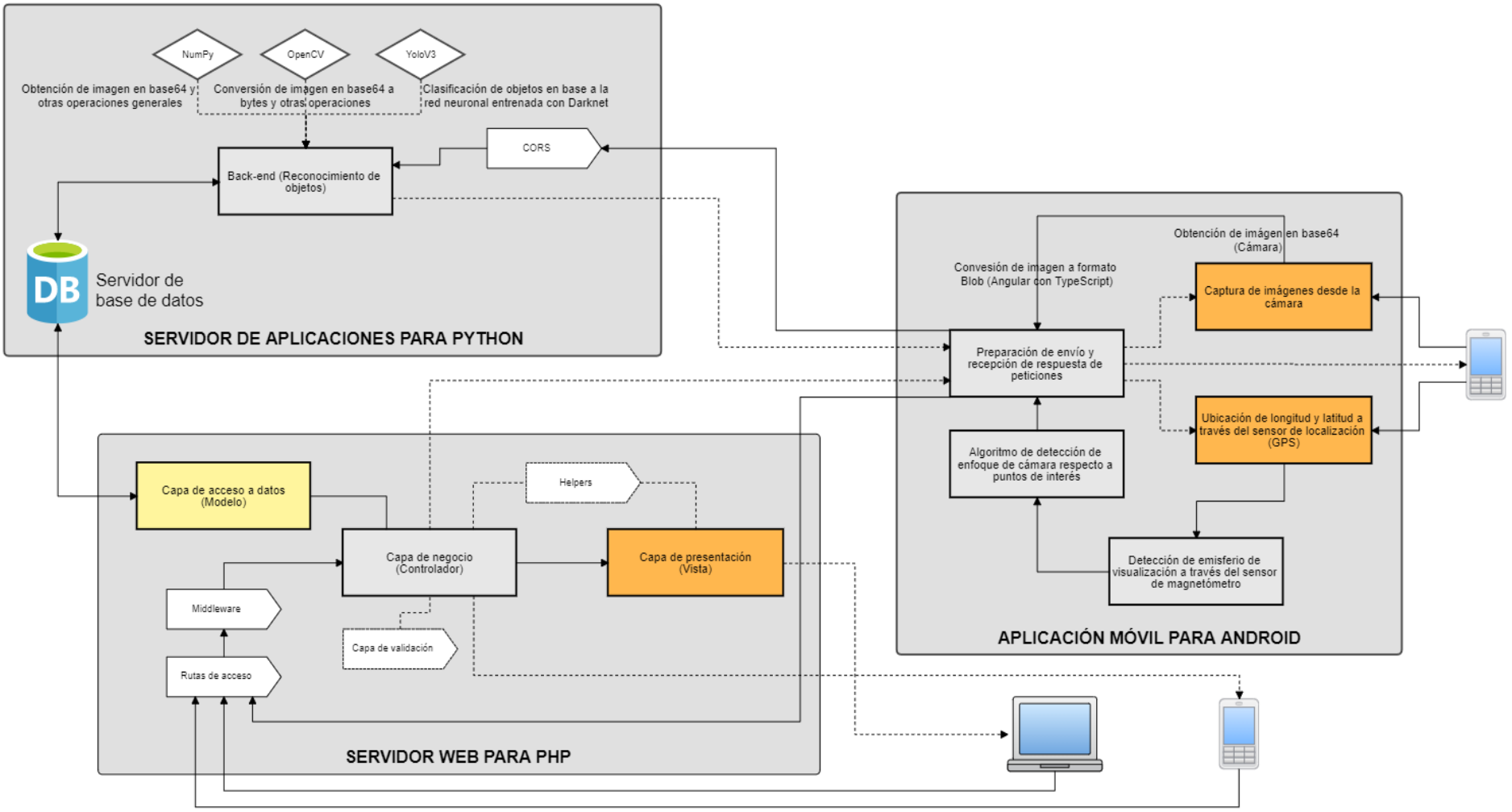


Figura 24. Arquitectura lógica de la implementación de los sistemas

A continuación, se documenta cada uno de los sistemas que conforman la arquitectura propuesta: i) Sistema de reconocimiento de recursos turísticos, ii) Sistema de gestión de contenido turístico y iii) Aplicación móvil de cliente final con realidad aumentada.

I. Sistema de reconocimiento de recursos turísticos

Definición general del sistema

Sistema en el que se desarrolló el algoritmo de reconocimiento de objetos turísticos y que es consumido como un servicio web desde la aplicación móvil desarrollada para Android. Constituye el corazón del proyecto de investigación. Este sistema permite el reconocimiento de objetos turísticos previamente cargados, su objetivo es lograr una precisión adecuada al reconocer un recurso turístico en tiempo real usando la conexión a internet. El algoritmo de reconocimiento se ejecuta en el lugar real del recurso turístico, respondiendo a la aplicación de usuario, la cual le envía imágenes capturadas por la cámara del móvil, permitiendo, entre un conjunto de elementos, reconocer el recurso para brindar información sobre este.

Especificación de requerimientos del sistema

El sistema se desarrolló de manera ágil, utilizando el marco de trabajo Scrum para el desarrollo de software, el cual privilegia el desarrollo antes que la documentación.

Los roles que se asignaron para el desarrollo del marco de trabajo se especifican en la Tabla 6.

Tabla 6

Roles del equipo Scrum del sistema de reconocimiento

Rol	Persona
Product Owner	Ronald Renteria
Scrum Master	Ronald Renteria
Development Team	Ronald Renteria

Los requerimientos del sistema fueron priorizados por el Product Owner y se definieron en el Product BackLog, tal como se muestra en la Tabla 7.

Tabla 7

Product BackLog del sistema de reconocimiento

Id	Historia de usuario (descripción)	Prioridad	Complejidad	Sprint	Duración (días)
01	Entrenar modelo de reconocimiento con *machine learning*	5	5	1	15
02	Evaluar la precisión del modelo entrenado	4	2	2	1
03	Reconocer los objetos aprendidos devolviendo su información turística relacionada	4	4	3	10

Donde: la prioridad y complejidad son valoradas desde 1 hasta 5, siendo 1 el menor y 5 el mayor valor.

A continuación, se presentan los Sprint BackLog cuya iteración completan el sistema.

Sprint BackLog 1

Identificador	SP01T01
Nombre	Captura y colección imágenes
Descripción	Definir los recursos turísticos que aprenderá el modelo y capturar para cada recurso la mayor cantidad de fotografías desde diversos ángulos, para construir el *dataset* de entrenamiento y validación.
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	SP01T02
Nombre	Preprocesamiento de imágenes
Descripción	Labor de elegir y depurar las imágenes capturadas que formarán el *dataset* final de entrenamiento.
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T03**
Nombre	Etiquetado de imágenes
Descripción	Tomar cada imagen del *dataset* de entrenamiento, delimitar o enmarcar el recurso turístico y asignarle su correspondiente etiqueta; utilizando el programa LabelImg_v1.8.0.
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T04**
Nombre	Configuración de Google Drive
Descripción	Crear en Drive la estructura de carpetas y archivos necesarios para que se pueda realizar el entrenamiento al acceder a estos desde Colab.
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T05**
Nombre	Creación de Notebook en Google Colab
Descripción	Crear *notebook* que realice: montar Drive como unidad, instalar controladores de CUDA, clonar la Darknet, copiar el *dataset* a Colab e iniciar el entrenamiento.
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Sprint BackLog 2

Identificador	**SP02T01**
Nombre	Validación del modelo
Descripción	Ejecutar los comandos de Darknet para obtener la precisión del archivo entrenado final yolov3_final.weights.
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	2

Sprint BackLog 3

Identificador	**SP03T01**
Nombre	Implementación de algoritmo de reconocimiento
Descripción	Realizar un algoritmo en Python, que permita recibir imágenes y determine con un nivel de certeza el reconocimiento de un objeto aprendido, accediendo a una base de datos para retornar información sobre el recurso reconocido.
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	3

Identificador	**SP03T02**
Nombre	Integración del algoritmo con el modelo entrenado
Descripción	Incluir el modelo entrenado, representado en el archivo yolov3_final.weights, dentro del proyecto de reconocimiento.
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	3

Identificador	SP03T03
Nombre	Exposición de funcionalidad como servicio web
Descripción	Empaquetar el proyecto de reconocimiento como una RESTful API para servir a una *app* móvil que le enviará imágenes.
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	3

Especificación de procedimientos del sistema

Dentro de los procedimientos de desarrollo se indica las herramientas utilizadas.

YOLO.- *You Only Look Once* o Solo miras una vez, es un algoritmo de detección de objetos que usa una única red neuronal a la imagen completa a ser evaluada, (Almog, 2020). Esta red divide la imagen en regiones y predice cuadros delimitadores y probabilidades para cada región. Estos cuadros delimitadores están ponderados por las probabilidades predichas. El modelo tiene varias ventajas sobre los sistemas basados en clasificadores. Evalúa la imagen completa en el momento de la prueba, por lo que sus predicciones se basan en el contexto global de la imagen. También hace predicciones con una única evaluación de red, a diferencia de sistemas como R-CNN, que requieren miles para una sola imagen. Esto lo hace extremadamente rápido, más de 1000 veces más rápido que R-CNN y 100 veces más rápido que Fast R-CNN, (Redmon & Farhadi, 2017). Por otro lado, aunque ya no es el algoritmo más preciso, es una muy buena opción cuando se necesita detección en tiempo real, sin perder demasiada precisión.

DarkNet.- Darknet es un marco de trabajo de red neuronal de código abierto escrito en C y CUDA. Admite cálculos de CPU y GPU. Darknet permite mayor velocidad que otras arquitecturas como NN o Faster RCNN, por ello, está escrita en C, (Redmon, 2016). El binomio Darknet y YOLO es un marco especializado, puesto que YOLO puede ejecutarse en la CPU, pero se obtiene 500 veces más velocidad en la GPU de Darknet ya que aprovecha CUDA y cuDNN. Este marco de aprendizaje profundo está escrito en C, pero una vez que entrena la red, no necesita la propia Darknet para la inferencia. OpenCV ha incorporado soporte para formatos Darknet, por lo que tanto el modelo como los pesos entrenados se pueden usar directamente en cualquier lugar donde se use OpenCV, también desde Python.

TensorFlow.- TensorFlow es una biblioteca de código abierto para el cálculo numérico y el aprendizaje automático a gran escala y agrupa una gran cantidad de modelos y algoritmos de aprendizaje automático y aprendizaje profundo o redes neuronales. Utiliza Python para proporcionar una API frontal conveniente para crear aplicaciones con el marco, mientras ejecuta esas aplicaciones en C++ de alto rendimiento. TensorFlow puede entrenar y ejecutar redes neuronales profundas para clasificación de dígitos escritos a mano, reconocimiento de imágenes, incrustaciones de palabras, redes neuronales recurrentes, modelos de secuencia a secuencia para traducción automática, procesamiento de lenguaje natural y simulaciones basadas en *Parcial Diferencial Ecuation* (ecuación diferencial parcial). TensorFlow admite la predicción de producción a escala, con los mismos modelos que se usan para el entrenamiento (Yegulalp, 2019).

LabelImg.- LabelImg es una herramienta de etiquetado de imágenes gráficas. Está escrito en Python y usa Qt para su interfaz gráfica. Los etiquetados se guardan como archivos XML en formato PASCAL VOC, el formato utilizado por ImageNet. Además, también es compatible con los formatos YOLO y CreateML (LabelImg, 2021).

Google Colaboratory.- Google Colaboratory o simplemente Google Colab es un entorno *notebook* Jupyter alojado que es de uso gratuito y no requiere configuración. Es un servicio en la nube gratuito y es compatible con GPU gratuita. Se usa para codificación del lenguaje de programación Python o para desarrollar aplicaciones de aprendizaje profundo utilizando bibliotecas populares como Keras, TensorFlow, PyTorch y OpenCV, pues todas ellas ya están instaladas. Por lo tanto, se podría decir que es un SaaS (Software as a Service) (Google, 2020).

CUDA.- *Compute Unified Device Architecture* es una plataforma de computación paralela y un modelo de programación desarrollado por la empresa Nvidia para computación general en sus propias GPU (unidades de procesamiento de gráficos). CUDA permite a los desarrolladores acelerar las aplicaciones de cómputo intensivo aprovechando la potencia de las GPU para paralelizar el cómputo.

CuDNN.- CUDA *Deep Neural Network* (cuDNN) es una biblioteca de primitivas acelerada por GPU para redes neuronales profundas. Proporciona implementaciones altamente ajustadas de rutinas que surgen con frecuencia en aplicaciones DNN.

Python.- Python es un lenguaje de programación interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipificación dinámica, tipos de datos

dinámicos de muy alto nivel y clases (Python Software Foundation, 2020). Python es un lenguaje de programación de uso general popular que se puede utilizar para una amplia variedad de aplicaciones. Incluye estructuras de datos de alto nivel, tipado dinámico, enlace dinámico y muchas más características que lo hacen útil para el desarrollo de aplicaciones complejas como para secuencias de comandos o "código de unión" que conecta componentes entre sí. También se puede ampliar para realizar llamadas al sistema a casi todos los sistemas operativos y ejecutar código escrito en C o C ++. Debido a su ubicuidad y capacidad para ejecutarse en casi todas las arquitecturas de sistemas, Python es un lenguaje universal que se encuentra en una variedad de aplicaciones diferentes, (Opensource.com, 2020). Se utilizó Python como lenguaje de programación en la parte del *back-end* para el procesamiento de imágenes y el posterior reconocimiento de objetos, de acuerdo al entrenamiento de la red neuronal artificial, bajo Darknet.

PyCharm.- PyCharm es uno de los IDE de Python más populares, está desarrollado por JetBrains. Tiene el soporte en su desarrollo web usando Django, está disponible como una aplicación multiplataforma y es compatible con las plataformas Linux, macOS y Windows. PyCharm viene con una gran cantidad de módulos, paquetes y herramientas para acelerar el desarrollo de Python al tiempo que reduce el esfuerzo requerido para hacer lo mismo en gran medida, simultáneamente (PyCharm, 2020).

Flask.- Es un “micro” *framework* para la creación de aplicaciones web con Python bajo el patrón MVC. Se utilizó por la facilidad de crear servicios web como API REST que expondrá los servicios de reconocimiento.

OpenCV.- Librería usada para la decodificación de imágenes, ya que estas fueron recibidas a través de objetos en base64, lo cual permitió convertir y redimensionar para un mayor rendimiento en el proceso de reconocimiento de objetos; así mismo, cabe mencionar que esta librería se usa ampliamente en el mismo proceso de reconocimiento, pero de manera transparente, ya que lo incorpora YOLOv3.

NumPy.- Librería usada para la manipulación de vectores y matrices (Usado parcialmente en el proceso de decodificación de las imágenes, conjuntamente con OpenCV). Cabe mencionar que al igual que OpenCV, esta librería es usada ampliamente en YOLOv3, y al igual que con OpenCV, su uso es de forma transparente.

REST.- REST (*REpresentational State Transfer*) o Transferencia de estado representativa, es un estilo arquitectónico para proporcionar estándares entre sistemas

informáticos en la web, lo que facilita la comunicación entre los sistemas. Los sistemas compatibles con REST, a menudo denominados sistemas RESTful, se caracterizan por la forma en que no tienen estado y separan las preocupaciones del cliente y del servidor (CodeAcademy, 2020). La arquitectura REST fue usada conjuntamente con el *framework* Flask, y se implementó una API REST, para la comunicación de punto a punto, para el procesamiento de las imágenes y reconocimiento y/o clasificación de los mismos.

CORS.- *Cross-origin resource sharing* (CORS), es un protocolo que permite que los scripts que se ejecutan en un cliente de navegador interactúen con recursos de un origen diferente. Esto es útil porque, gracias a la política del mismo origen seguida por XMLHttpRequest y fetch, JavaScript solo puede realizar llamadas a URL que residen en el mismo origen que la ubicación donde se ejecuta el *script*, (Hobbs, 2019). Se utilizó para el intercambio de recursos (Peticiones HTTP) entre diferentes dominios de la aplicación.

Postman.- Es una plataforma de colaboración para el desarrollo de API's. Simplifica cada paso de la creación de una API y optimiza la colaboración para crear mejores API y más rápido (Postman, 2021). Por lo que se utilizó esta plataforma para probar si la API de reconocimiento creada en Python retorna lo deseado.

Resultados

Se logró organizar la información necesaria para que Darknet realice el entrenamiento en Google Drive como se observa en la Figura 25 y se creó el notebook en Google Colab llamado Train_with_Darknet_and_GPU.ipynb que se muestra en el Anexo 1, con las instrucciones necesarias para realizar el entrenamiento y validación.

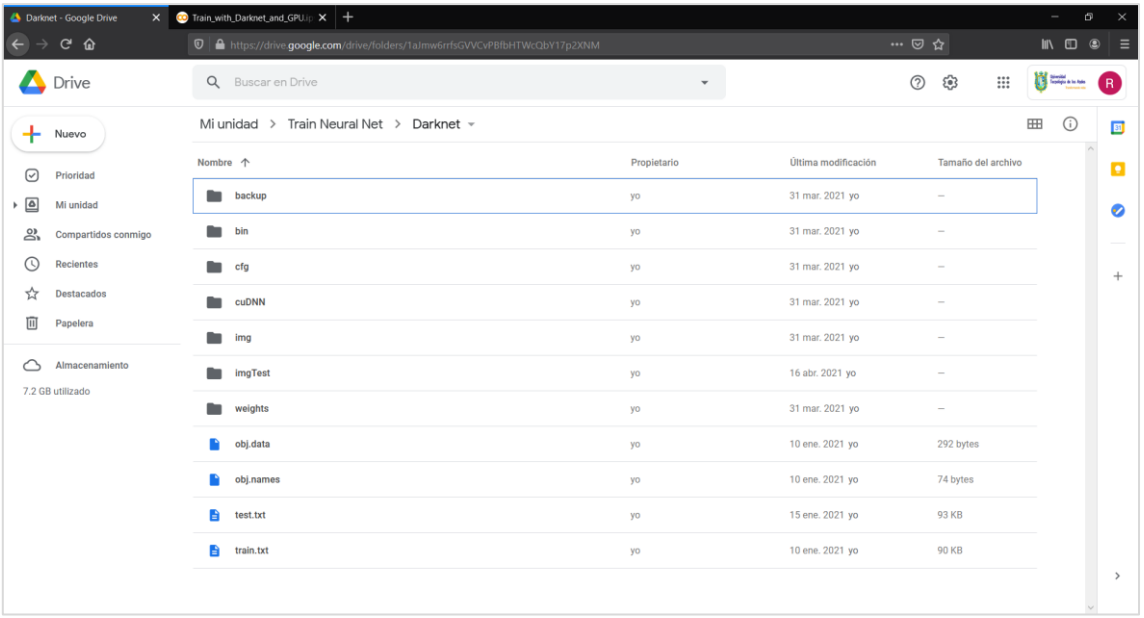


Figura 25. Organización de archivos y carpetas en Google Drive

Posteriormente se realizó la validación del modelo (Figura 26) dentro de Darknet dando los siguientes resultados.

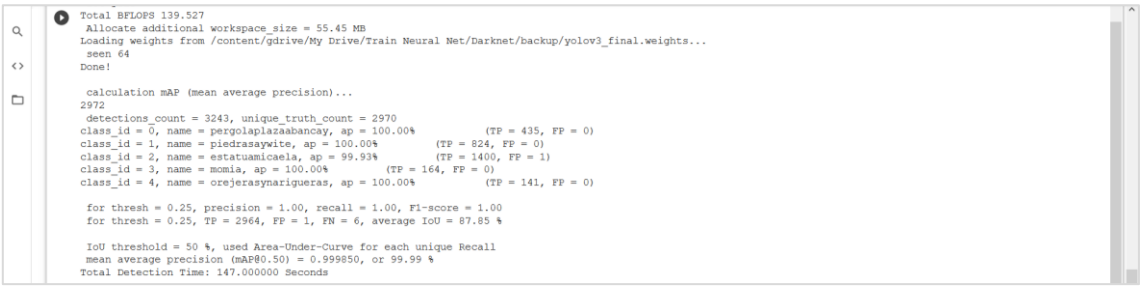


Figura 26. Resultados de validación del modelo

A continuación, se elaboró el algoritmo de reconocimiento en el IDE Pycharm, en el lenguaje Python con el *framework* Flask, como se observa en la Figura 27. El archivo resultante se llamó objectRecognition.py y se muestra completo en el Anexo 2.

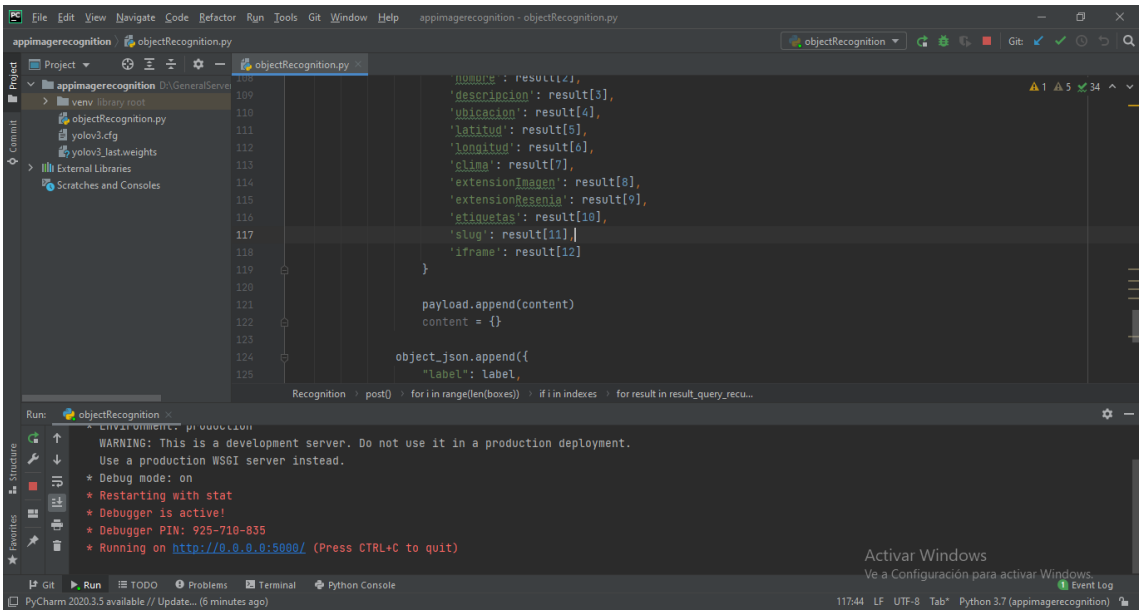


Figura 27. Implementación de sistema de reconocimiento

Finalmente, se probó el API hecho en Python realizando peticiones con Postman para comprobar los resultados obtenidos (Figura 28).

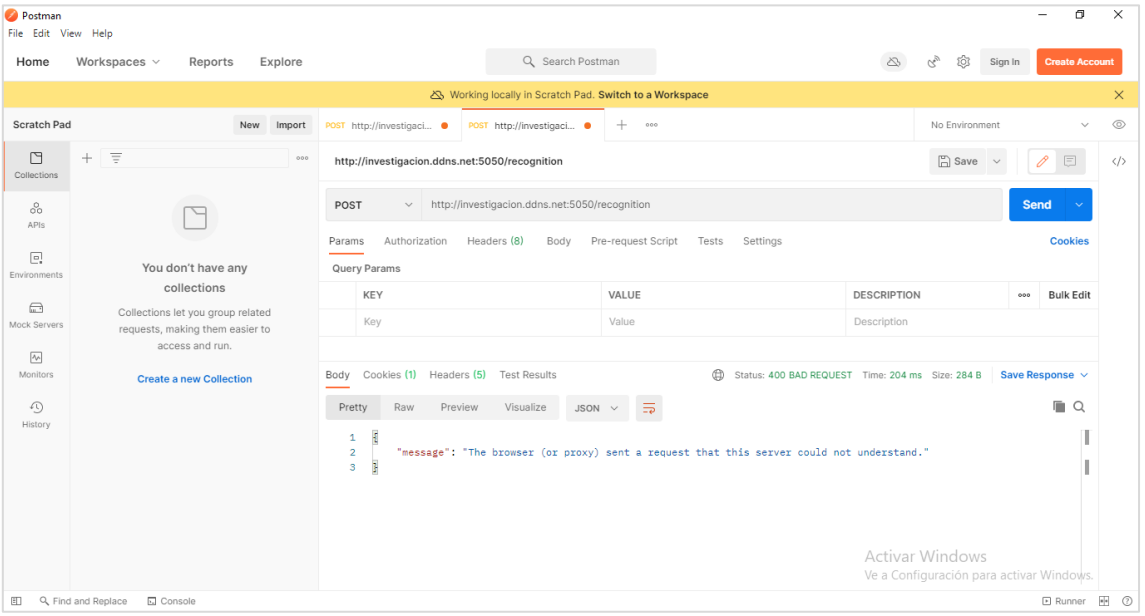


Figura 28. Prueba de la API de reconocimiento

II. Sistema de gestión de contenido turístico

Definición general del sistema

Es un sistema de información con enfoque al usuario administrativo, permite la carga de información que se desplegará posteriormente al reconocimiento del recurso turístico.

En este sistema se gestiona la inserción, modificación, edición y eliminación de archivos multimedia, enlaces, puntos de interés. En referencia a los puntos de interés se trabaja con la geolocalización por medio de la API de Google Maps.

Especificación de requerimientos del sistema

De la misma manera se utilizó el *framework* Scrum para desarrollar el sistema.

La Tabla 8 muestra los roles asignados para el desarrollo con el marco de trabajo.

Tabla 8

Roles del equipo Scrum del sistema de gestión de contenido

Rol	Persona
Product Owner	Ronald Renteria
Scrum Master	Ronald Renteria
Development Team	Ronald Renteria

Los requerimientos del sistema fueron priorizados por el Product Owner y se definieron en el Product BackLog, tal como se muestra en la Tabla 9.

Tabla 9

Product BackLog del sistema de gestión de contenido

Id	Historia de usuario (descripción)	Priori-dad	Com-plejidad	Sprint	Dura-ción (días)
01	Construcción de la arquitectura	5	3	1	15
02	Gestión de usuarios	4	2	2	5
03	Gestión de categorías	4	2	3	3
04	Gestión de recursos turísticos	3	3	4	10
05	Gestión de puntos de interés turístico	3	3	5	10

Donde: la prioridad y complejidad son valoradas desde 1 hasta 5, siendo 1 el menor y 5 el mayor valor.

A continuación, se presentan los Sprint BackLog cuya iteración completan el sistema.

Sprint BackLog 1

Identificador	**SP01T01**
Nombre	Diseño de la arquitectura de la aplicación
Descripción	Elaborar en coordinación con el product owner la arquitectura de la aplicación que satisfaga los requerimientos
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T02**
Nombre	Diseño del modelo de datos
Descripción	Elaborar la base de datos relacional con la información proporcionada por el product owner, identificando las entidades que controlan los usuarios, categorías, recursos y puntos de interés turístico
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T03**
Nombre	Configuración del *framework* de desarrollo
Descripción	Instalar y preparar el marco de desarrollo Laravel para el inicio del desarrollo del sistema
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Sprint BackLog 2

Identificador	**SP02T01**
Nombre	Gestión de usuarios
Descripción	Crear, modificar, listar y eliminar información de usuarios del sistema
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	2

Identificador	**SP02T02**
Nombre	Autenticación de usuarios
Descripción	Acceder al sistema utilizando un formulario de autenticación y gestionar la sesión del usuario
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	2

Sprint BackLog 3

Identificador	**SP03T01**
Nombre	Gestión de categorías
Descripción	Crear, modificar, listar y eliminar información de categorías
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	3

Sprint BackLog 4

Identificador	**SP04T01**
Nombre	Gestión de recursos turísticos
Descripción	Crear, modificar, listar y eliminar información de recursos turísticos
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	4

Sprint BackLog 5

Identificador	**SP05T01**
Nombre	Gestión de puntos de interés turístico
Descripción	Crear, modificar, listar y eliminar información de puntos de interés turístico
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	5

Especificaciones de procedimientos del sistema

Dentro de los procedimientos de desarrollo se indica las herramientas utilizadas.

PHP.- Lenguaje que se usó para toda la gestión del *back-end*, del sistema desarrollado para la gestión de contenido turístico.

MySQL.- Sistema de gestión de base de datos que se usó para el almacenamiento de todo el contenido ingresado para el uso de los sistemas desarrollados.

Laravel.- Marco de trabajo para la creación de aplicaciones web en el lenguaje de programación PHP de forma elegante y simple y se caracteriza por la aplicación del patrón Modelo Vista Controlador (MVC).

JavaScript.- Lenguaje de programación que se usó para la interactividad del usuario con los formularios web.

JQuery.- Librería que se usó para potenciar y agilizar el desarrollo con JavaScript.

HTML y CSS.- Lenguajes que se usaron para la maquetación y modelamiento web.

Bootstrap.- *Framework* que se usó para el manejo de componentes web, simplificando así el desarrollo nativo de estos.

API de Google Maps.- API que se usó para la interactividad del usuario, con un mapa virtual, para la ubicación y posiciones geográficas (Fue usado con mucha prioridad, para definir los puntos de interés respecto a la ubicación de un usuario que esté interactuando con su dispositivo móvil).

FormValidation.io.- Librería que se usó para el sistema de validación de datos en el *front-end*, sobre todos los formularios del sistema web.

MySQL Workbench.- Herramienta de diseño de base de datos que se usó específicamente para la ejecución de los *scripts* creados para MySQL.

REST.- Arquitectura para aplicaciones basadas en la red, la cual fue usada para la comunicación de punto a punto, para la obtención de datos de un recurso turístico o de un punto de interés.

CORS.- Mecanismo que se usó para el intercambio de recursos (Peticiones HTTP) entre diferentes dominios de la aplicación.

Resultados

Se creó la base de datos en el sistema gestor MySQL, la cual se muestra en la Figura 29, la cual se llamó dbartour.

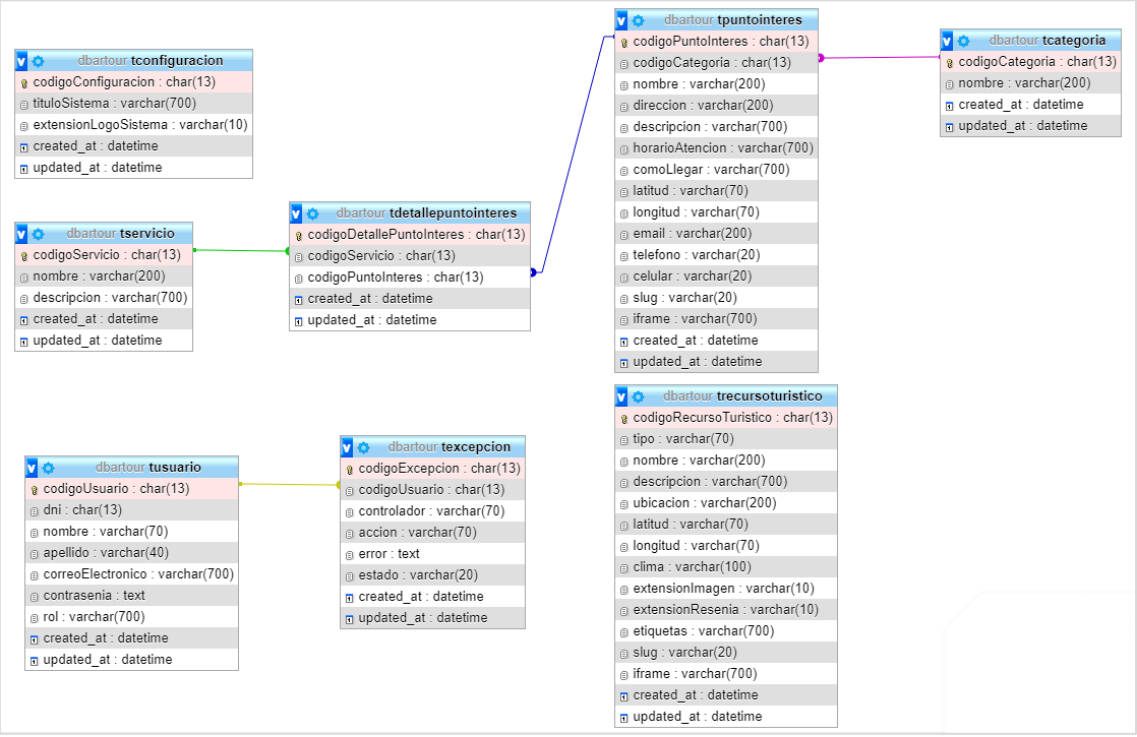


Figura 29. Modelo relacional de la base de datos

Luego se instaló el *framework* Laravel versión 5.5 para la construcción del sistema, cuya estructura de carpetas y archivos se observa en la Figura 30.

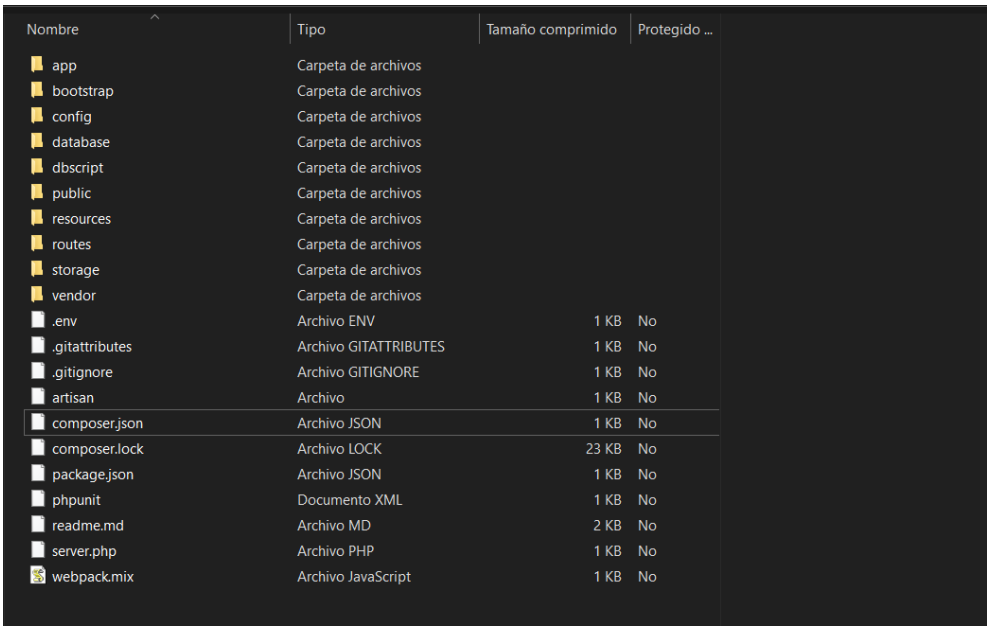


Figura 30. Estructura de carpetas Laravel

Con el *framework* configurado se procedió a realizar la gestión de usuarios, lo cual es facilitado por Laravel. Se muestran capturas de la gestión de usuarios (Figura 31), de autenticación y sesión (Figura 32).

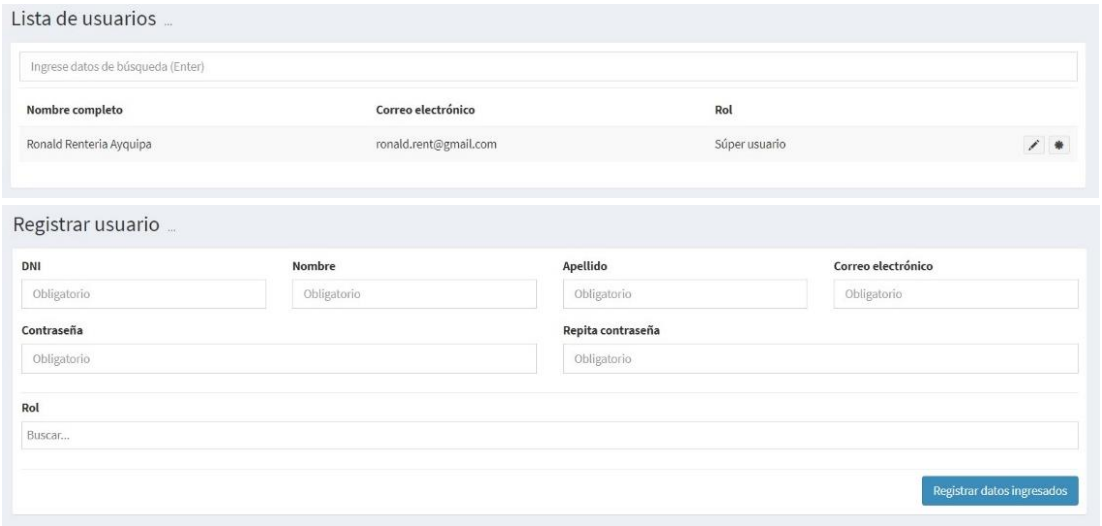


Figura 31. Capturas de pantalla gestión de usuarios

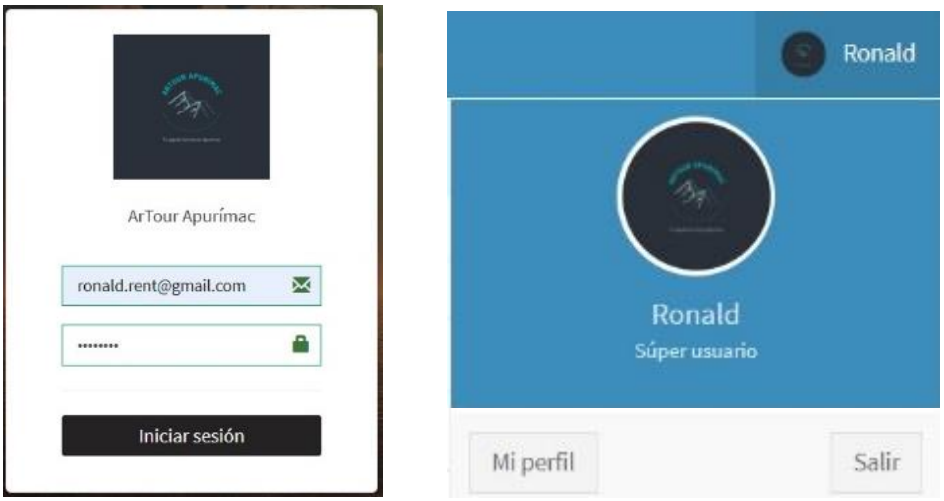


Figura 32. Capturas de pantalla de autenticación y sesión

Se realizó la gestión de categorías turísticas para clasificar tantos los recursos como puntos de interés turístico (Figura 33).

Figura 33. Capturas de pantalla gestión de categorías

De la misma manera se realizó la gestión de recursos turísticos para administrar la información que posteriormente será mostrada al detectar o reconocer un recurso con la aplicación de usuario final, como se observa en la Figura 34.

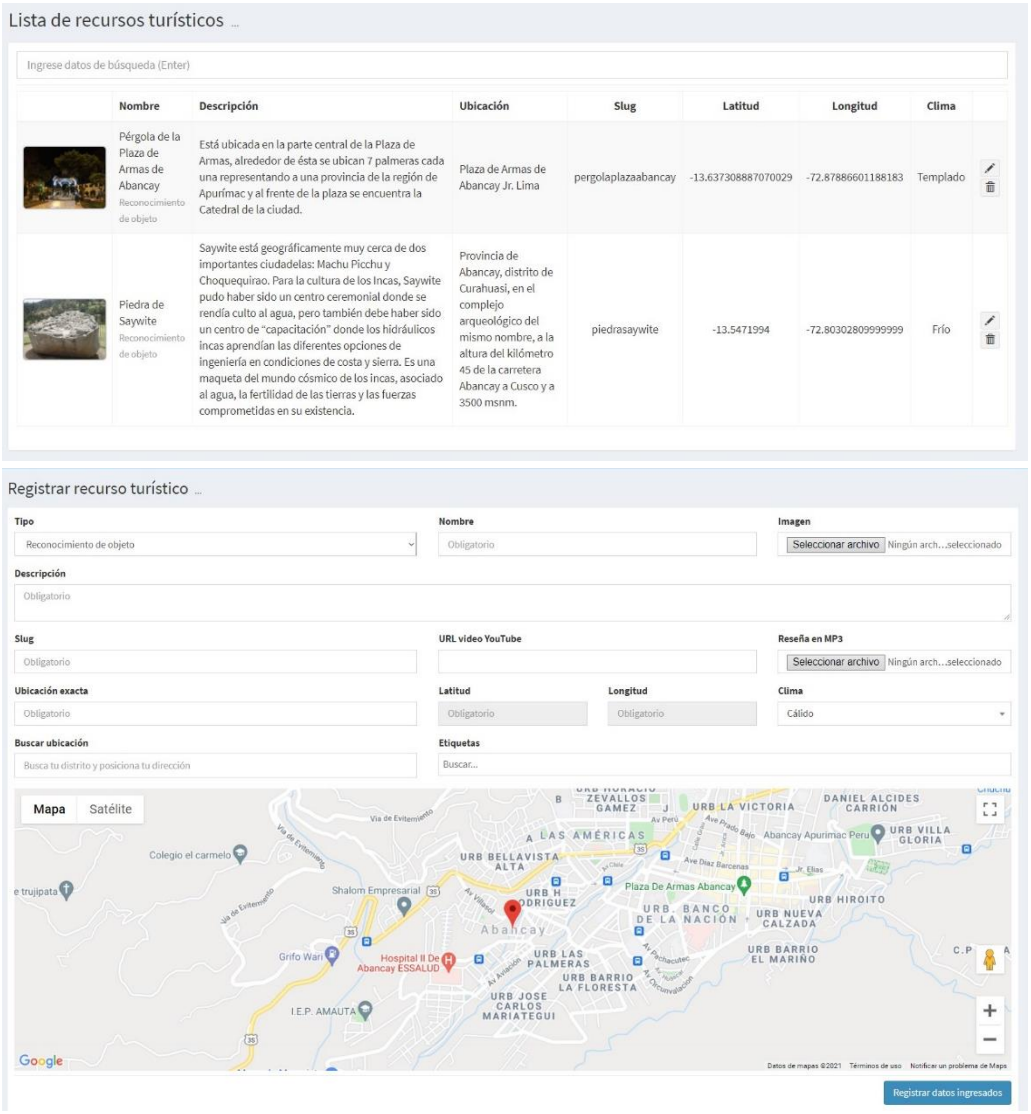


Figura 34. Capturas de pantalla gestión de recursos turísticos

Finalmente se realizó la gestión de los puntos de interés turístico con la información necesaria para ser mostrada cuando sea consumida por la app móvil de usuario final (ver Figura 35).

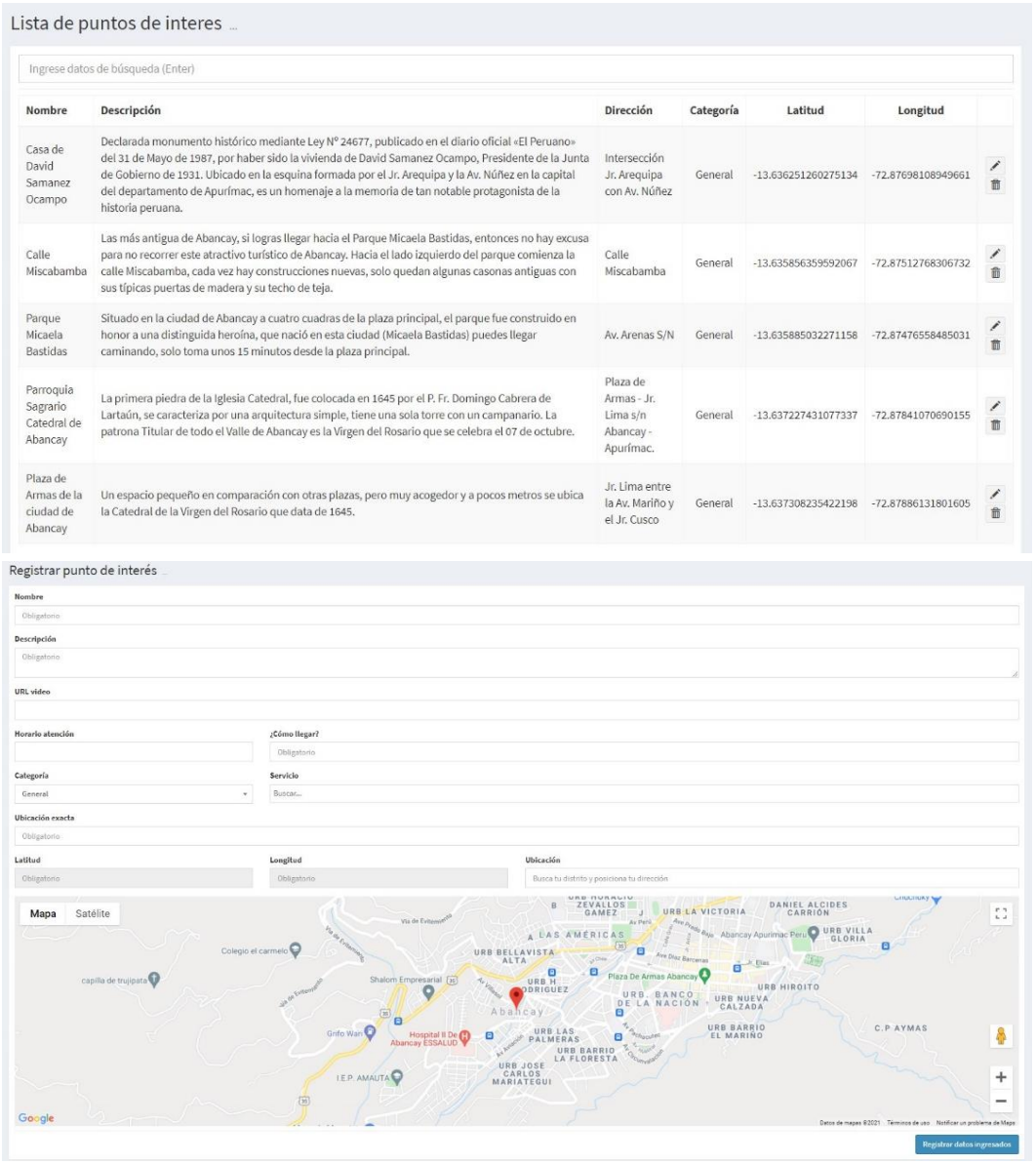


Figura 35. Capturas de pantalla gestión de puntos de interés turístico

III. Aplicación móvil de cliente final con realidad aumentada

Definición general del sistema

Aplicación a través de la cual se consume: i) el servicio de reconocimiento de objetos: este es el servicio principal, que permite al usuario apuntar su cámara del celular para reconocer el recurso turístico en tiempo real y ii) la información de los puntos de interés: una vez, el recurso ha sido reconocido, se muestran dos tipos de información basados en algoritmos acorde a la necesidad de detección y son: información de los objetos o recursos turísticos, es decir, información multimedia acerca del lugar, e información de posiciones respecto a la ubicación del usuario, esto es, que al geolocalizar y conocer el rumbo del usuario, la aplicación presenta información sobre puntos de interés alrededor del usuario

en un radio determinado. Toda esta información se despliega en tiempo real y mediante realidad aumentada.

Especificación de requerimientos del sistema

De la misma manera se utilizó el *framework* Scrum para desarrollar el sistema.

La Tabla 10 muestra los roles asignados para el desarrollo con el marco de trabajo.

Tabla 10

Roles del equipo Scrum de la aplicación móvil de cliente final

Rol	Persona
Product Owner	Ronald Renteria
Scrum Master	Ronald Renteria
Development Team	Ronald Renteria

Los requerimientos del sistema fueron priorizados por el Product Owner y se definieron en el Product BackLog, tal como se muestra en la Tabla 11.

Tabla 11

Product BackLog de la aplicación móvil de cliente final

Id	Historia de usuario (descripción)	Prioridad	Complejidad	Sprint	Duración (días)
01	Reconocimiento de recursos turísticos y muestra de información mediante realidad aumentada	5	5	1	30
02	Localización de puntos de interés turístico cercanos y muestra de información mediante realidad aumentada	5	5	2	30

Donde: la prioridad y complejidad son valoradas desde 1 hasta 5, siendo 1 el menor y 5 el mayor valor.

A continuación, se presentan los Sprint BackLog cuya iteración completan el sistema.

Sprint BackLog 1

Identificador	**SP01T01**
Nombre	Diseño de navegación de la aplicación móvil
Descripción	Diseñar las interfaces e interacción con la aplicación móvil de realidad aumentada
Prioridad	5
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T02**
Nombre	Listado de recursos turísticos cargados para reconocimiento
Descripción	Dar la opción de visualizar los recursos turísticos que han sido entrenados en el modelo de *machine learning* y que pueden ser reconocidos por la aplicación
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	**SP01T03**
Nombre	Reconocimiento de recurso turístico
Descripción	Tomar imágenes con la cámara del móvil para ser enviadas a la API de reconocimiento de objetos, la cual devolverá la información del recurso si fue reconocido en la imagen enviada o no devolverá nada si en la imagen no encontró un recurso entrenado
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Identificador	SP01T04
Nombre	Despliegue de información turística mediante realidad aumentada
Descripción	Al recibir la información del recurso detectado, esta se debe mostrar en la pantalla del móvil a través de opciones (texto, audio, video)
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	1

Sprint BackLog 2

Identificador	SP02T01
Nombre	Configuración del radio de búsqueda de puntos de interés
Descripción	Definir el radio (en km) para la búsqueda de puntos de interés turístico desde la posición real del móvil
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	2

Identificador	SP02T02
Nombre	Localización de puntos de interés turístico
Descripción	De acuerdo a la geolocalización proporcionada por el GPS, al rumbo proporcionado por al magnetómetro del móvil y al radio de búsqueda configurado, se debe mostrar en la pantalla del móvil los puntos de interés turístico con indicadores de posición (izquierda, derecha o centro) de acuerdo a su orientación; los puntos de interés que estén en el centro muestran un botón para poder visualizar su infografía.
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	2

Identificador	SP02T03
Nombre	Despliegue de información turística mediante realidad aumentada
Descripción	Al solicitar información de un punto de interés turístico, se envía la petición al sistema de gestión de contenido turístico, la cual devuelve la información almacenada en la base de datos, esta información se debe mostrar en la pantalla del móvil a través de opciones (texto, audio, video)
Prioridad	4
Estado	Terminado
Usuario	Ronald Renteria
Iteración	2

Especificaciones de procedimientos del sistema

Dentro de los procedimientos de desarrollo se indica las herramientas utilizadas.

Ionic.- SDK para el desarrollo de aplicaciones móviles híbridas, que utiliza HTML, CSS y JavaScript, el cual se usó para desarrollar la aplicación móvil para el usuario final.

JavaScript.- Lenguaje de programación usado como base para todo el desarrollo de la aplicación móvil.

Angular.- *Framework* de JavaScript que se usó para el desarrollo de todo el *back-end* de la aplicación móvil.

Apache Cordova.- Entorno de desarrollo de aplicaciones móviles multiplataforma utilizando tecnologías web estándares. Trabaja por debajo de Ionic para realizar algunas tareas como acceder a sensores del móvil.

HttpClient.- Librería que se usó para el consumo de los servicios REST expuestos por el sistema de reconocimiento de objetos y el sistema de gestión de contenido.

Sensor de localización (GPS).- Sensor del dispositivo móvil que se usó para ubicar las coordenadas donde se encontraba el usuario final, para la posterior ubicación de este, respecto a ciertos puntos de interés a cierta distancia en kilómetros a la redonda.

Sensor magnetómetro.- Sensor del dispositivo móvil que se usó para la detección de la orientación del usuario final, respecto a los puntos de interés y al enfoque de la cámara (dirección de visualización).

Cámara del móvil.- Dispositivo usado para el enfoque de objetos y detección de enfoque de puntos de interés.

Resultados

Se crearon las interfaces para el usuario final, definiendo un sencillo sistema de navegación lineal en dos idiomas español e inglés, las cuales se observan en la Figura 36.

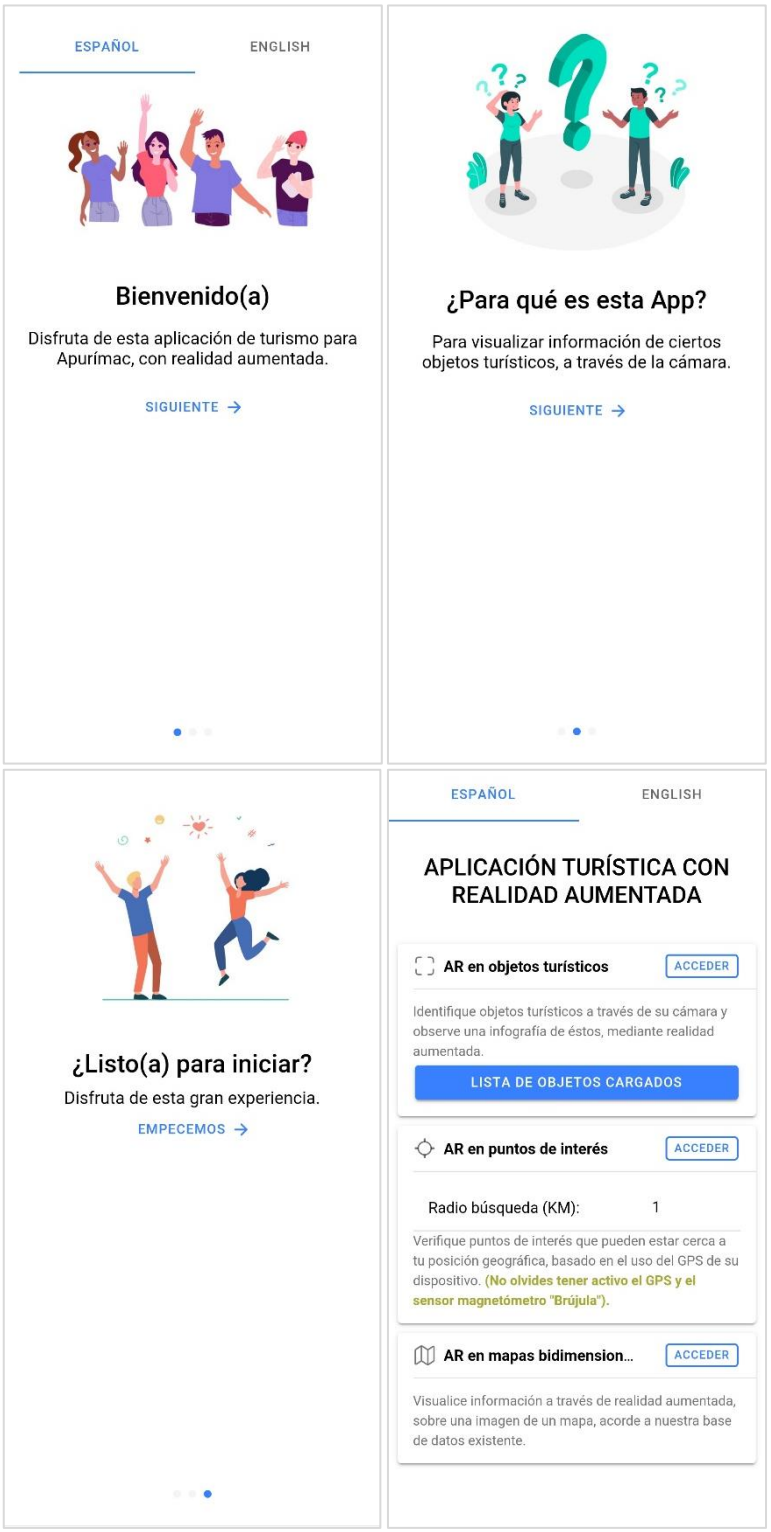


Figura 36. Interfaz de la app móvil

Se permitió listar los recursos turísticos que la *app* puede reconocer y que han sido entrenados previamente en la red neuronal (Figura 37).

Figura 37. Lista de recursos turísticos entrenados

A continuación, se implementó el reconocimiento de objetos, los cuales una vez reconocidos la *app* indicaba su reconocimiento con una vibración y el mensaje "objeto encontrado", además de opciones para poder visualizar contenido turístico relacionado (*Figura 38*).

Figura 38. Indicación de objeto reconocido

Cuando el usuario elige ver contenido relacionado se puede ver este contenido mediante realidad aumentada, este contenido puede ser texto (*Figura 39*), o puede ser multimedia (*Figura 40*).

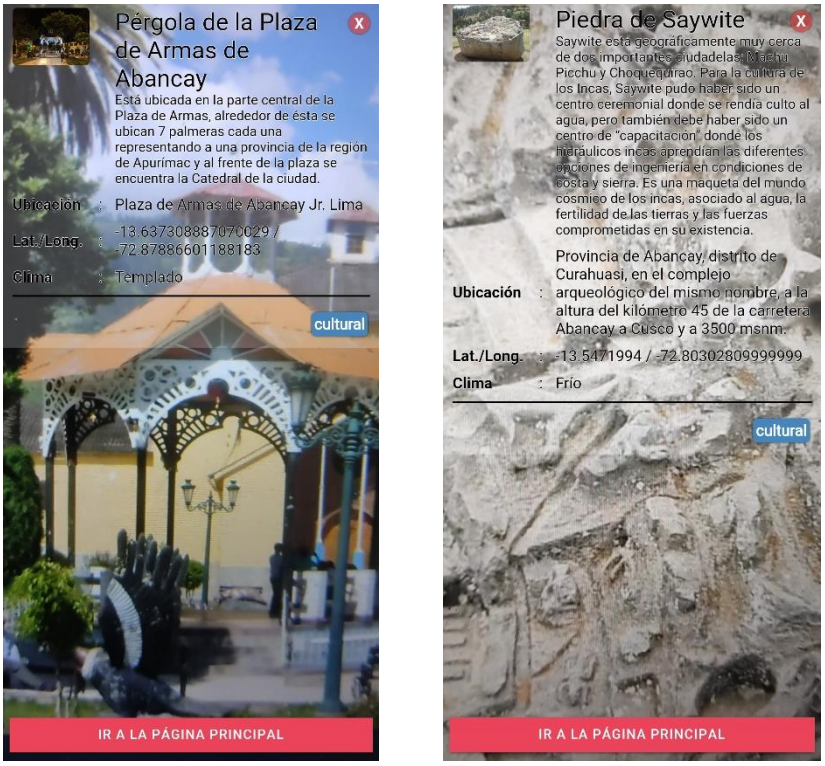


Figura 39. Muestra de texto mediante realidad aumentada

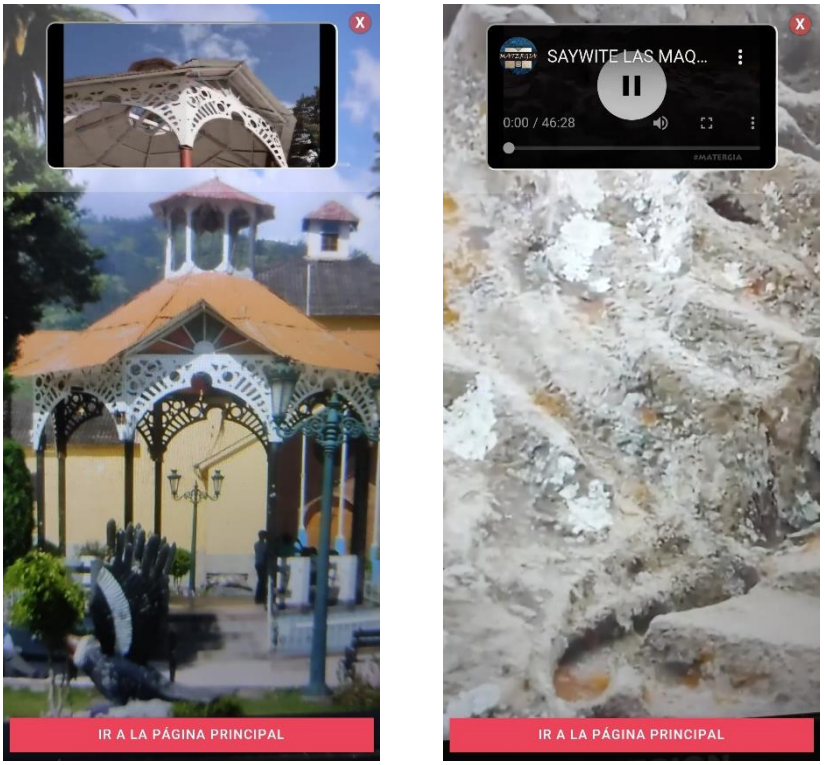


Figura 40. Muestra de video mediante realidad aumentada

Además, para mostrar los puntos de interés turístico se permite configurar el radio de búsqueda entre 1-9 km, tal como se aprecia en la *Figura 41*.

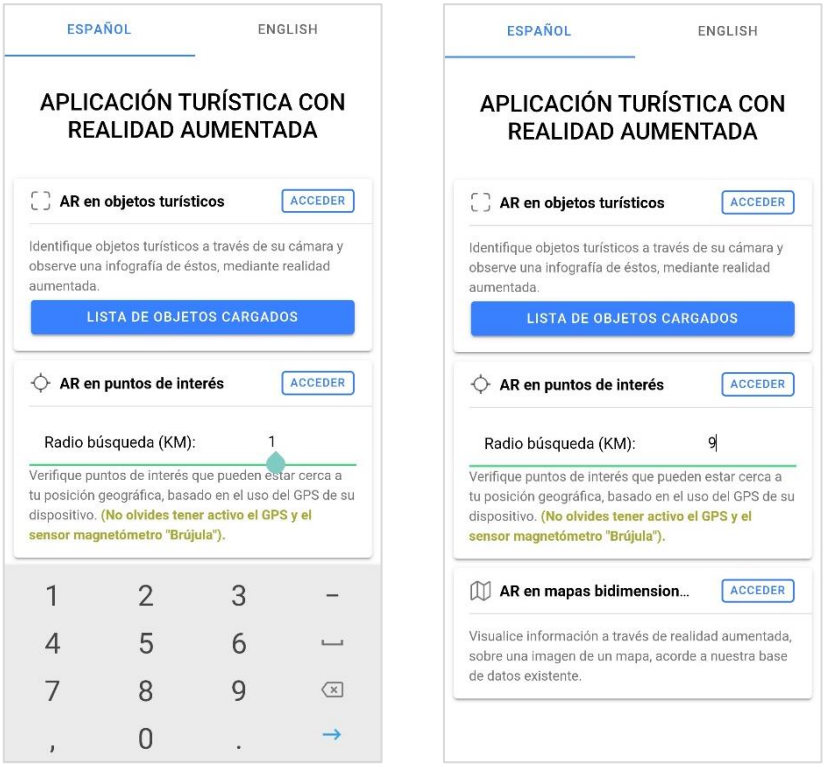


Figura 41. Configuración del radio de búsqueda

Configurado el radio de búsqueda se permite acceder a esta funcionalidad y visualizar los puntos de interés de acuerdo a su posición respecto a la ubicación y orientación (rumbo) del móvil, ver *Figura 42*.

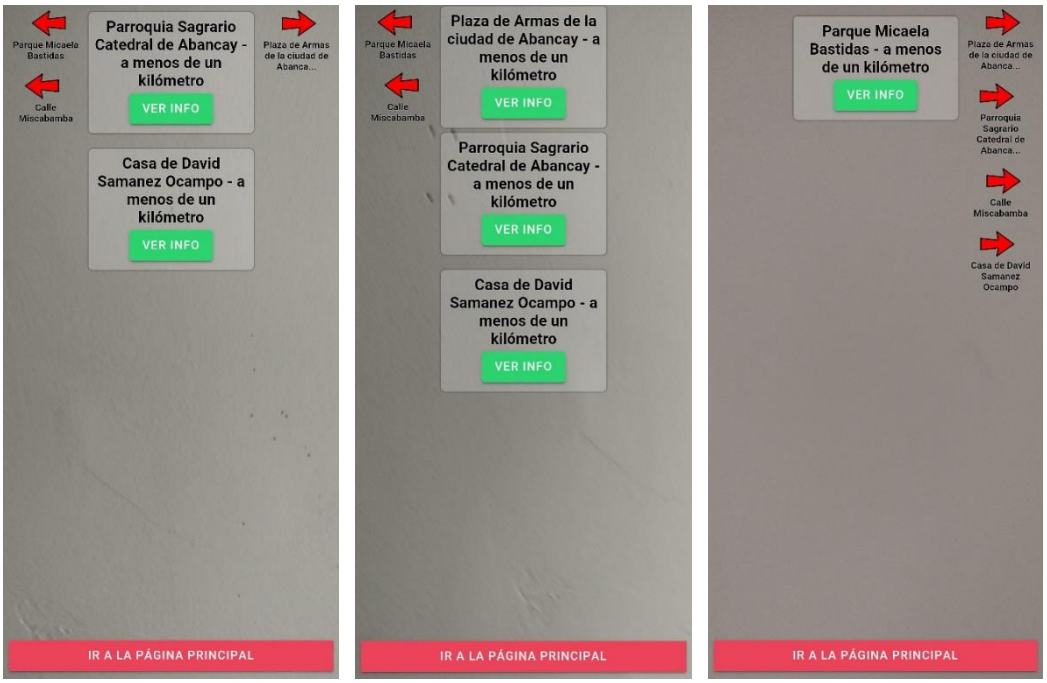


Figura 42. Localización de puntos de interés

Los puntos de interés hacia donde se orienta el móvil pueden ser seleccionados para visualizar su contenido relacionado, esto se aprecia en la *Figura 43*.

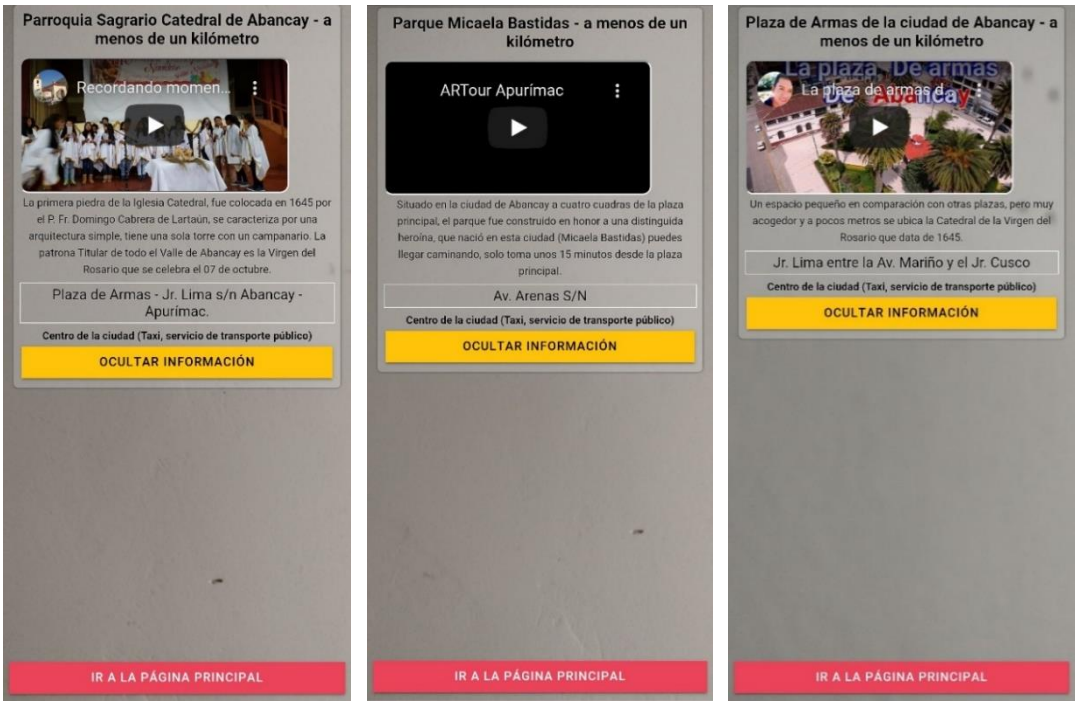


Figura 43. Despliegue de información mediante realidad aumentada

El archivo APK de la *app*, fue instalado y probado en los equipos móviles Xiaomi Mi A2 Lite y POCO X3 NFC, los cuales tienen los sensores GPS y magnetómetro (brújula) necesarios para la utilización de la app (*Figura 44*).

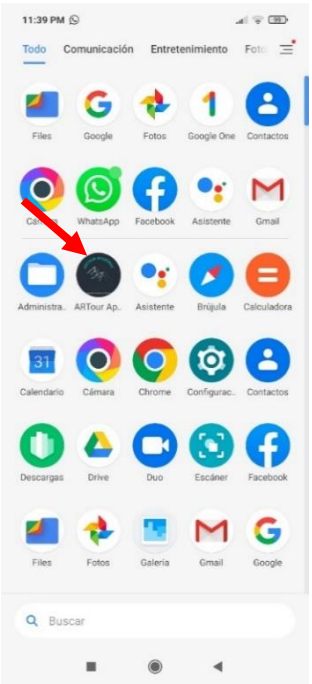


Figura 44. App móvil instalada

4.4 Discusión de resultados

A partir de los hallazgos encontrados en la investigación, se acepta la hipótesis que establece que al aplicar técnicas de *machine learning* se logra un nivel de precisión adecuado del modelo entrenado para reconocer recursos turísticos en Apurímac.

Esto guarda relación con los estudios de X. Wang *et al.* (2020) que diseñaron una red neuronal convolucional de 19 capas sobre TensorFlow, y experimentalmente demostraron que los modelos de redes convolucionales presentados son superiores a otros modelos de redes neuronales en precisión y eficiencia.

Así también, en la investigación de Quintero *et al.* (2018) en la cual usan algoritmos de *Deep Learning* mediante redes neuronales convolucionales en TensorFlow y Keras con arquitectura Inception v3, para reconocer macroinvertebrados en agua, obteniendo una precisión por encima del 90% en los resultados obtenidos en la clasificación de imágenes.

En el artículo de Lou *et al.* (2020), los cuales implementaron una red neuronal convolucional con arquitectura Inception v3 para el reconocimiento de escenas; la cual se entrenó con los conjuntos de datos PLACES2, SUN397 Y NUS-WIDE, obteniendo precisiones de 89.3, 94.6 y 85.7% respectivamente, concluyendo que el modelo es mejor para grandes conjuntos de imágenes.

También Cobeña *et al.* (2019) utilizaron una red neuronal convolucional para reconocer imágenes satelitales de cultivos de caña de azúcar, la cual mostró un promedio de predicción de 94.16%.

Traore *et al.* (2018) clasificaron patógenos epidémicos mediante una red neuronal convolucional, obteniendo una precisión en la clasificación del 94%, lo cual servirá para hacer más inteligentes los microscopios futuros.

Lawal *et al.* (2019) diseñaron una red neuronal convolucional en MATLAB para el reconocimiento de género basado en rostros de nigerianos, logrando una precisión de reconocimiento general de 98.72%. Igualmente Xie *et al.* (2019) proponen un modelo para reconocimiento facial en una red neuronal convolucional basada en Keras sobre Python utilizando el *dataset* Olivettifaces y modificando parámetros para su optimización, con lo que obtuvieron una tasa de reconocimiento del 97.5%.

En lo que respecta al algoritmo YOLO, Dewantoro *et al.* (2020) utilizaron este algoritmo para detectar el número de vehículos en una carretera, concluyendo que si se utiliza un *dataset* propio el algoritmo da mejores resultados. Sumit *et al.* (2020) compararon YOLO y Mask R-CNN en el reconocimiento de figuras humanas con *dataset* propios, resultando que el desempeño de YOLO fue superior, así mismo YOLO realiza la detección en menor tiempo. Cheng (2020) también comparó R-CNN y YOLO en sus versiones 1, 2 y 3; en la identificación de imágenes, aplicados a la detección de texto con un conjunto de datos propio, resultando que YOLO obtiene una precisión de 73.54% en un tiempo de 0.105 segundos mientras que R-CNN logra 76.29% en un tiempo de 6.9 segundos; concluyendo que, si bien la precisión no varía significativamente entre ambos, existe una clara diferencia entre los tiempos de ejecución, siendo YOLO el de desempeño superior.

Pero hay que tener en cuenta el estudio de Hu *et al.* (2020) que señalan que el procesamiento de imágenes basado en redes neuronales convolucionales aún no tiene una teoría formal completa, debido a que los parámetros de configuración de la red como el tamaño del filtro y la profundidad de la red, deben ser definidos por experimentación, lo que limita la universalidad de la estructura de red.

CONCLUSIONES

Una vez concluido el estudio y tras la realización del análisis, interpretación y discusión de los resultados, se llega a las siguientes conclusiones:

- Para lograr el reconocimiento de recursos turísticos, se implementaron redes neuronales convolucionales con YOLOv3, el modelo resultante se evaluó de diferentes formas para obtener su precisión, primero se obtuvo la precisión dentro de Darknet con el conjunto de imágenes de prueba, luego se sometió a evaluaciones de precisión dentro de Darknet con imágenes de diferentes fuentes como la web y archivos fotográficos y finalmente se evaluó la precisión utilizando el modelo mediante una *app* móvil de manera presencial en cada recurso turístico. El resultado de las mediciones evidenció un nivel de precisión en el reconocimiento de los recursos turísticos superior a 90%.
- Se desarrolló una *app* móvil con realidad aumentada con la finalidad de determinar en tiempo real puntos de interés turístico cercanos en base a la geolocalización y orientación de la cámara móvil, utilizando fórmulas de la geometría y trigonometría esférica para encontrar la distancia entre dos puntos sobre una superficie esférica y el rumbo o ángulo de orientación, esto permitió detectar los puntos de interés turístico en un radio de "*x*" km a la redonda y mostrarlos con indicadores de acuerdo a su posicionamiento -izquierda, derecha o centro- de acuerdo al rumbo del móvil. Esta *app* se probó en diferentes puntos de la ciudad para evaluar su eficacia, dando como resultado que la proporción de aciertos de la *app* móvil identificando correctamente los puntos de interés turístico cercanos en tiempo real es superior a 0.9, con lo que se concluye que la *app* móvil identifica correctamente los puntos de interés turístico cercanos en tiempo real.
- Se propuso una arquitectura que logra la interoperabilidad de distintas tecnologías tales como *machine learning* para el entrenamiento y detección de objetos (recursos turísticos), web para el sistema de gestión de contenidos turísticos, móvil para la *app* de usuario final y realidad aumentada para mostrar la información turística en la pantalla del móvil mostrando como fondo el enfoque de la cámara.

RECOMENDACIONES

En base a la delimitación del estudio, además de las limitantes y restricciones presentadas, las recomendaciones propuestas del estudio son las siguientes:

- En esta investigación se realizó un entrenamiento de la red neuronal convolucional YOLOv3 en un entorno *cloud* con GPU gratuita que nos brinda Google Colaboratory, lo cual causó que el entrenamiento se lleve en varios días debido a que Colab solo permite uso de GPU continua por 12 horas, esto provocaba que el entrenamiento se pausara cumplido el tiempo. Por lo que se recomienda realizar este mismo proceso en un sistema local con GPU(s) para tener una máquina dedicada al entrenamiento, esto implicaría la adquisición de una computadora con capacidades para *machine learning*.
- La red neuronal convolucional utilizada es un *fork* de YOLOv3 especialmente acondicionada para Google Colab, por lo que se sugiere experimentar con otras CNN, como por ejemplo la versión Tiny YOLOv3 que es un modelo muy pequeño para entornos restringidos o la versión 4 de YOLO, incluso la versión en desarrollo YOLOv5.
- Por motivos de la pandemia global, hubo restricciones de movilización y no se pudo realizar el entrenamiento del modelo con recursos turísticos de provincias más alejadas del departamento. Por lo que se recomienda y queda pendiente que se realice un estudio con elección aleatoria de los recursos turísticos y de acuerdo a la población.
- Durante la realización del estudio se percibió que, para hacer la detección de recursos turísticos, además de la técnica de *machine learning*, también podría realizarse con geolocalización del móvil y ángulo de su cámara, lo cual puede ser un tema de futuros estudios.
- Para no tener tantos sitios que visitar o mostrar en la pantalla del móvil, se puede hacer un sistema de recomendaciones que le muestre al turista una cantidad limitada de puntos de interés turístico, de acuerdo a su perfil, en vez de mostrarle una inmensa cantidad de estos.
- Otra de las limitaciones que provocó en esta investigación la pandemia, cuarentena y consecuente inmovilización social, es que la actividad turística fue anulada, por lo que no se pudo realizar pruebas de uso del sistema por los turistas y promotores de turismo. Esto impidió tener estadísticas de uso de la aplicación, así como obtener la retroalimentación necesaria para la mejora del sistema.

BIBLIOGRAFÍA

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, *22*(10), 1533–1545. https://doi.org/10.1109/TASLP.2014.2339736

Afsahhosseini, F., & Al-mulla, Y. (2020). *Machine Learning in Tourism*. (December). https://doi.org/10.1145/3426826.3426837

Allwin, M. S., Tanoto, A., Forbes, W., & Ashari. (2019). Forecasting Determination by Ousing Development Schedule Using Learning Machine Approach Using Clustering Method. *Journal of Physics: Conference Series*, *1230*(1). https://doi.org/10.1088/1742-6596/1230/1/012019

Almog, U. (2020). YOLO V3 Explained. Retrieved from https://towardsdatascience.com/yolo-v3-explained-ff5b850390f

Amato, G., Falchi, F., & Rabitti, F. (2011). Landmark recognition in VISITO Tuscany. *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, 1–2. https://doi.org/10.1007/978-3-642-27978-2_1

Arnal, M. G. (2018). *Estudio y aplicación de las redes neuronales convolucionales 3D*. Barcelona.

Barrile, V., Fotia, A., Ponterio, R., Mollica-Nardo, V., Giuffrida, D., & Mastelloni, M. (2019). A combined study of art works preserved in the archaeological museums: 3d survey, spectroscopic approach and augmented reality. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W11*, 201–207.

Breen, K. (2020). What is Augmented Reality? How does it work? Let's see! Retrieved July 9, 2020, from https://www.immersiv.io/blog/what-is-augmented-reality-definition/

Burkapalli, V., & Patil, P. (2019). Food image segmentation using edge adaptive based deep-CNNs. *International Journal of Intelligent Unmanned Systems*, *8*(4), 243–252. https://doi.org/10.1108/IJIUS-09-2019-0053

Caballero, V. A., & Villacorta, A. E. (2014). *Aplicación móvil basada en Realidad Aumentada para promocionar los principales atractivos turísticos y restaurantes*

calificados del Centro Histórico de Lima. Universidad de San Martín de Porres, Lima.

Callejas, M., Quiroga, J. G., & Alarcón, A. C. (2011). Ambiente interactivo para visualizar sitios turísticos, mediante realidad aumentada implementando Layar. *Ciencia e Ingeniería Neogranadina*, *XXI*(2), 91–105.

Carreton, A. (2017). ¿Qué es la puesta en valor del patrimonio cultural? Retrieved from Patrimonio inteligente website: https://patrimoniointeligente.com/la-puesta-en-valor-del-patrimonio-cultural/

Cavallaro, R., Hybinette, M., White, M., & Balch, T. (2011). Augmenting live broadcast sports with 3D tracking information. *IEEE Multimedia 2011*, *18*(4), 38–47.

Chawla, S. (2019). 7 Successful Applications of AI & Machine Learning in the Travel Industry. Retrieved January 4, 2021, from Hackernoon website: https://hackernoon.com/successful-implications-of-ai-machine-learning-in-travel-industry-3040f3e1d48c

Cheng, R. (2020). A survey: Comparison between Convolutional Neural Network and YOLO in image identification. *Journal of Physics: Conference Series*, *1453*(1). https://doi.org/10.1088/1742-6596/1453/1/012139

Cobeña, J. P., Atiencia, J. M., & Andryshchenko, I. S. (2019). Convolutional neural network in the recognition of spatial images of sugarcane crops in the troncal region of the coast of Ecuador. *Procedia Computer Science*, *150*, 757–763. https://doi.org/10.1016/j.procs.2019.02.001

CodeAcademy. (2020). What is REST? Learn about how to design web services using the REST paradigm. Retrieved from https://www.codecademy.com/articles/what-is-rest

Cox, C. R., Moscardini, E. H., Cohen, A. S., & Tucker, R. P. (2020). Machine learning for suicidology: A practical review of exploratory and hypothesis-driven approaches. *Clinical Psychology Review*, *82*(October), 101940. https://doi.org/10.1016/j.cpr.2020.101940

Craig, A. B. (2013). *Understanding Augmented Reality: Concepts and Applications*. Elsevier.

Cruña, J. (2019). ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador. Retrieved from https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

DDCA. (2018). *Reporte de Visitas al Sitio Arqueológico de Saywite*. Abancay.

Dewantoro, N., Fernando, P. N., & Tan, S. (2020). YOLO Algorithm Accuracy Analysis in Detecting Amount of Vehicles at the Intersection. *IOP Conference Series: Earth and Environmental Science*, *426*(1). https://doi.org/10.1088/1755-1315/426/1/012164

Dini, G., & Mura, M. D. (2015). Application of Augmented Reality Techniques in Through-life Engineering Services. *Procedia CIRP*, *38*, 14–23. https://doi.org/10.1016/j.procir.2015.07.044

Fernández, M. T., & Cuadrado, R. (2014). El impacto de las nuevas tecnologías en el sector turístico: Aplicación de la realidad aumentada al turismo cultural. *International Journal of World of Tourism*, *1*(2), 11–18. https://doi.org/10.12795/ijwt.2014.i02.02

Garaas, T., Xiao, M., & Pomplun, M. (2007). Personalized Spell Checking Using Neural Networks. *Posters*, *0000*, 559–563. Retrieved from http://www.cs.umb.edu/~marc/pubs/garaas_xiao_pomplun_HCII2007.pdf

Goodfellow, I., Bemgop, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from https://www.deeplearningbook.org/contents/intro.html

Google, I. (2020). What is Colaboratory? Retrieved from https://colab.research.google.com/notebooks/intro.ipynb

Gurney, K. (1997). *An Introduction to Neural Networks* (U. of Sheffield, Ed.). Taylor & Francis Group.

Hernández, R., Fernandez, C., & Baptista, P. (2010). *Metodología de la investigación* (Jesús Mare). México.

Hobbs, S. (2019). CORS Tutorial: A Guide to Cross-Origin Resource Sharing. Retrieved from https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/

Hu, H., Lyu, J., & Yin, X. (2020). Research and Prospect of Image Recognition Based

on Convolutional Neural Network. *Journal of Physics: Conference Series*, *1574*(1), 1–7. https://doi.org/10.1088/1742-6596/1574/1/012161

ICOMOS, A. (2013). *The Burra Charter: The Australia ICOMOS Charter for places of cultural Significance*.

Jin, W. (2020). Research on Machine Learning and Its Algorithms and Development. *Journal of Physics: Conference Series*, *1544*(1). https://doi.org/10.1088/1742-6596/1544/1/012003

Kaufmann, M. (1983). *Machine Learning: An Artificial Intelligence Approach*. Palo Alto - California: Tioga Publisher Co.

Kim, Y. (2017). An analysis of convolutional neural networks for sentence classification. *2017 43rd Latin American Computer Conference, CLEI 2017*, *2017-Janua*, 1–5. https://doi.org/10.1109/CLEI.2017.8226381

LabelImg. (2021). LabelImg. Retrieved from https://github.com/tzutalin/labelImg

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, 260–270. https://doi.org/10.18653/v1/n16-1030

Lawal, C. O., Akinrinmade, A. A., & Badejo, J. A. (2019). Face-based Gender recognition Analysis for Nigerians Using CNN. *Journal of Physics: Conference Series*, *1378*(3). https://doi.org/10.1088/1742-6596/1378/3/032014

Lawson, C., Martí, J. M., Radivojevic, T., Jonnalagadda, S. V. R., Gentz, R., Hillson, N., … Martin, H. G. (2019). Machine Learning for Metabolic Engineering: a Review . *Metabolic Engineering*, (October). https://doi.org/10.1016/j.ymben.2020.10.005

Lou, G., & Shi, H. (2020). Face Image Recognition Based on Convolutional Neural Network. *ARTIFICIAL INTELLIGENCE (AI)-DRIVEN SPECTRUM MANAGEMENT*, 117–124.

Manuri, F., & Sanna, A. (2016). A Survey on Applications of Augmented Reality. *Advances in Computer Science: An International Journal*, *5*(1), 18–27. Retrieved from http://www.acsij.org/acsij/article/view/400/350

Missinglink.ai. (2018). The Complete Guide to Artificial Neural Networks: Concepts and Models. Retrieved December 4, 2020, from https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/

Moreno, M. (2019). *App de asistencia en exteriores a personas con discapacidad visual bajo cualquier condición de luz*. Universidad de Alicante.

Opensource.com. (2020). What is Python? Retrieved from https://opensource.com/resources/python

Parvez, M. O. (2020). Use of machine learning technology for tourist and organizational services: high-tech innovation in the hospitality industry. *Journal of Tourism Futures*. Retrieved from https://www.emerald.com/insight/content/doi/10.1108/JTF-09-2019-0083/full/html

Pierdicca, R., Paolanti, M., & Frontoni, E. (2019). eTourism: ICT and its role for tourism management. *Journal of Hospitality and Tourism Technology*.

Postman. (2021). Postman. Retrieved from https://www.postman.com

PyCharm. (2020). IDE de Python para desarrolladores profesionales. Retrieved from https://www.jetbrains.com/es-es/pycharm/

Python Software Foundation, F. (2020). General Python FAQ. Retrieved December 27, 2020, from https://docs.python.org/3/faq/general.html#general-information

Qiu, C., Zhou, S., Liu, Z., Gao, Q., & Tan, J. (2019). Digital assembly technology based on augmented reality and digital twins: a review. *Virtual Reality & Intelligent Hardware*, *1*(6), 597–610. https://doi.org/10.1016/j.vrih.2019.10.002

Quintero, C., Merchán, F., Cornejo, A., & Galán, J. S. (2018). Uso de Redes Neuronales Convolucionales para el Reconocimiento Automático de Imágenes de Macroinvertebrados para el Biomonitoreo Participativo. *KnE Engineering*, *3*(1), 585. https://doi.org/10.18502/keg.v3i1.1462

Redmon, J. (2016). Darknet: Open Source Neural Networks in C. Retrieved from https://pjreddie.com/darknet/

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*,

2017-Janua, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

Sharma, A., & Dipti, C. (2013). Character recognition using neural networks. *11th IEEE International Symposium on Computational Intelligence and Informatics, CINTI 2010 - Proceedings*, *4*(April), 663–667. https://doi.org/10.1109/CINTI.2010.5672225

SIGMincetur. (2021). Sistema de Información Georeferencial del Mincetur. Retrieved February 19, 2021, from https://sigmincetur.mincetur.gob.pe/turismo/

Silva, E. (2020). Entrenamiento de la Red Neuronal Convolucional YOLO para objetos propios. Retrieved from https://biorobotics.fi-p.unam.mx/wp-content/uploads/Courses/reconocimiento_de_patrones/tutoriales/YOLO-Introducción-e-implementación-.pdf

SUBDERE. (2012). *Puesta en valor del patrimonio*.

Sumit, S. S., Watada, J., Roy, A., & Rambli, D. R. A. (2020). In object detection deep learning methods, YOLO shows supremum to Mask R-CNN. *Journal of Physics: Conference Series*, *1529*(4). https://doi.org/10.1088/1742-6596/1529/4/042086

Traore, B. B., Kamsu-Foguem, B., & Tangara, F. (2018). Deep convolution neural network for image recognition. *Ecological Informatics*, *48*, 257–268. https://doi.org/10.1016/j.ecoinf.2018.10.002

Vargas, Z. R. (2009). *La investigación aplicada: Una forma de conocer las realidades con evidencia científica*.

Vorraber, W., Gasser, J., Webb, H., Neubacher, D., & Url, P. (2020). Assessing augmented reality in production: Remote-assisted maintenance with HoloLens. *Procedia CIRP*, *88*, 139–144. https://doi.org/10.1016/j.procir.2020.05.025

Wang, C. (2011). Application of Virtual Reality Technology in Digital Tourism. *Third International Conference on Multimedia Information Networking and Security*, 537–541.

Wang, P., Qian, Y., Soong, F. K., He, L., & Zhao, H. (2015). *Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network*. Retrieved from http://arxiv.org/abs/1510.06168

Wang, X., & Yang, Z. (2020). Research on Classification and Recognition of Object

Image Based on Convolutional Neural Network. *IOP Conference Series: Materials Science and Engineering*, *782*(4). https://doi.org/10.1088/1757-899X/782/4/042062

Wangchuk, K., Riyamongkol, P., & Waranusast, R. (2020). Real-time Bhutanese Sign Language digits recognition system using Convolutional Neural Network. *ICT Express*, (xxxx). https://doi.org/10.1016/j.icte.2020.08.002

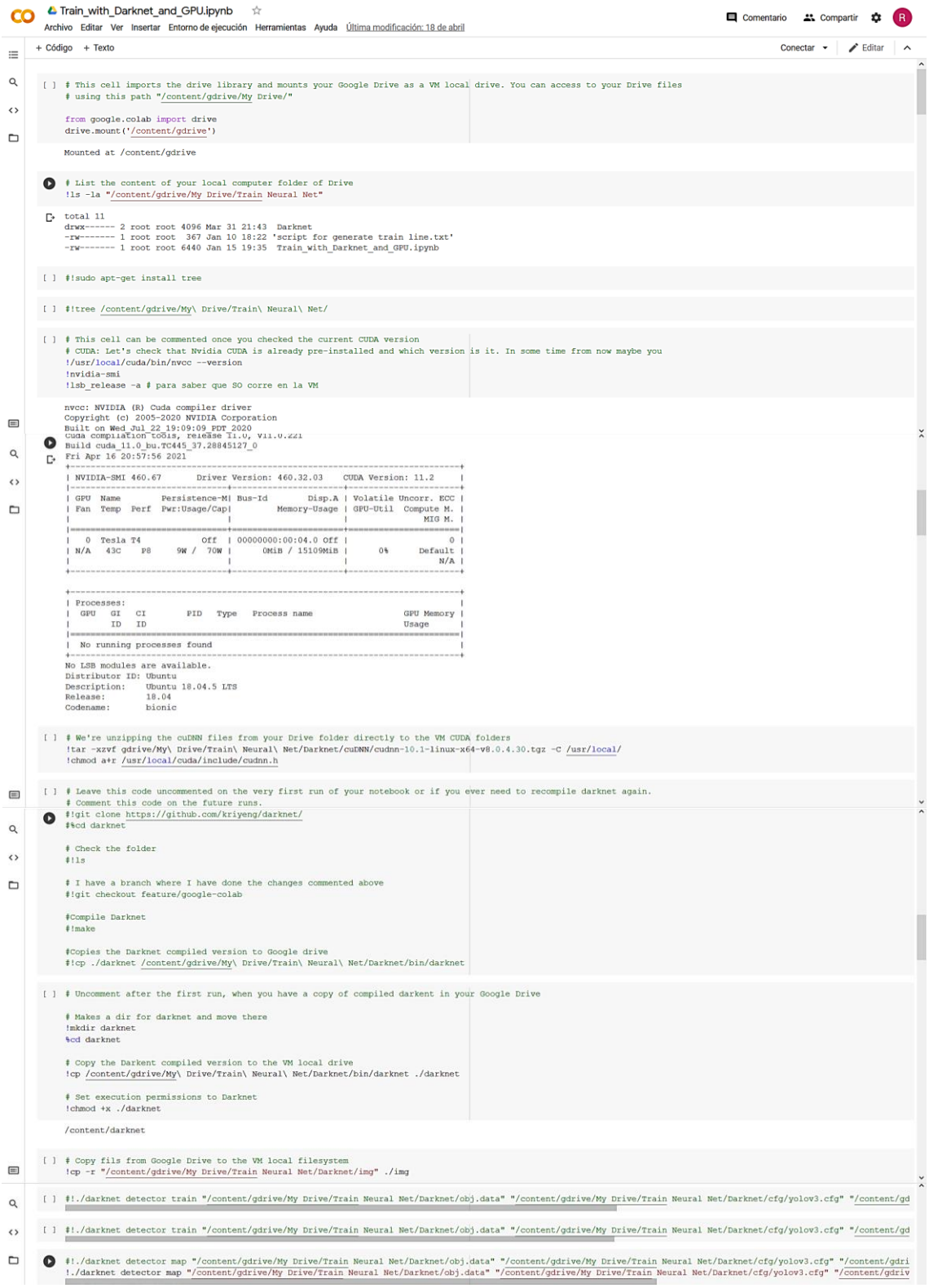
Xie, Z., Li, J., & Shi, H. (2019). A Face Recognition Method Based on CNN. *Journal of Physics: Conference Series*, *1395*(1). https://doi.org/10.1088/1742-6596/1395/1/012006

Yegulalp, S. (2019). What is TensorFlow? The machine learning library explained. Retrieved June 18, 2019, from https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

Zherdev, D. A. (2017). Object recognition of real targets using modelled SAR images. *Journal of Physics: Conference Series*, *936*(1). https://doi.org/10.1088/1742-6596/936/1/012076

ANEXOS

Anexo 1. Notebook de Colab para el entrenamiento



Train_with_Darknet_and_GPU.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Última modificación: 18 de abril

Comentario Compartir

+ Código + Texto

Conectar Editar

```
# This cell imports the drive library and mounts your Google Drive as a VM local drive. You can access to your Drive files
# using this path "/content/gdrive/My Drive/"

from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
# List the content of your local computer folder of Drive
!ls -la "/content/gdrive/My Drive/Train Neural Net"
```

```
total 11
drwx------ 2 root root 4096 Mar 31 21:43  Darknet
-rw------- 1 root root  367 Jan 10 18:22 'script for generate train line.txt'
-rw------- 1 root root 6440 Jan 15 19:35  Train_with_Darknet_and_GPU.ipynb
```

```
#!sudo apt-get install tree
```

```
#!tree /content/gdrive/My\ Drive/Train\ Neural\ Net/
```

```
# This cell can be commented once you checked the current CUDA version
# CUDA: Let's check that Nvidia CUDA is already pre-installed and which version is it. In some time from now maybe you
!/usr/local/cuda/bin/nvcc --version
!nvidia-smi
!lsb_release -a # para saber que SO corre en la VM
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:09_PDT_2020
Cuda compilation tools, release 11.0, V11.0.221
Build cuda_11.0_bu.TC445_37.28845127_0
Fri Apr 16 20:57:56 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.67       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   43C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.5 LTS
Release:        18.04
Codename:       bionic
```

```
# We're unzipping the cuDNN files from your Drive folder directly to the VM CUDA folders
!tar -xzvf gdrive/My\ Drive/Train\ Neural\ Net/Darknet/cuDNN/cudnn-10.1-linux-x64-v8.0.4.30.tgz -C /usr/local/
!chmod a+r /usr/local/cuda/include/cudnn.h
```

```
# Leave this code uncommented on the very first run of your notebook or if you ever need to recompile darknet again.
# Comment this code on the future runs.
#!git clone https://github.com/kriyeng/darknet/
#%cd darknet

# Check the folder
#!ls

# I have a branch where I have done the changes commented above
#!git checkout feature/google-colab

#Compile Darknet
#!make

#Copies the Darknet compiled version to Google drive
#!cp ./darknet /content/gdrive/My\ Drive/Train\ Neural\ Net/Darknet/bin/darknet
```

```
# Uncomment after the first run, when you have a copy of compiled darkent in your Google Drive

# Makes a dir for darknet and move there
!mkdir darknet
%cd darknet

# Copy the Darkent compiled version to the VM local drive
!cp /content/gdrive/My\ Drive/Train\ Neural\ Net/Darknet/bin/darknet ./darknet

# Set execution permissions to Darknet
!chmod +x ./darknet
```

```
/content/darknet
```

```
# Copy fils from Google Drive to the VM local filesystem
!cp -r "/content/gdrive/My Drive/Train Neural Net/Darknet/img" ./img
```

```
#!./darknet detector train "/content/gdrive/My Drive/Train Neural Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neural Net/Darknet/cfg/yolov3.cfg" "/content/gd
```

```
#!./darknet detector train "/content/gdrive/My Drive/Train Neural Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neural Net/Darknet/cfg/yolov3.cfg" "/content/gd
```

```
#!./darknet detector map "/content/gdrive/My Drive/Train Neural Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neural Net/Darknet/cfg/yolov3.cfg" "/content/gdri
!./darknet detector map "/content/gdrive/My Drive/Train Neural Net/Darknet/obj.data" "/content/gdrive/My Drive/Train Neural Net/Darknet/cfg/yolov3.cfg" "/content/gdriv
```

Anexo 2. Código Python de reconocimiento (objectRecognition.py)

```
from flask import Flask, request
from flask_restful import Resource, Api
from flask_cors import CORS
from flask_mysqldb import MySQL
from flask_restful.representations import json

import cv2
import numpy as np

app = Flask(__name__)
cors = CORS(app, resources={r"/*": {"origins": "*"}})

api = Api(app)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '030191'
app.config['MYSQL_DB'] = 'dbartour'

mysql = MySQL(app)

net = cv2.dnn.readNet('yolov3_last.weights', 'yolov3.cfg')

classes = [
    'pergolaplazaabancay',
    'piedrasaywite',
    'estatuamicaela',
    'momia',
    'orejerasynarigueras'
]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOut-
Layers()]


class Recognition(Resource):

    @classmethod
    def post(cls):
        object_json = []

        file_temp = request.files['buffer_image'].read()
        request_type = request.form['request_type']

        npimg = np.frombuffer(file_temp, np.uint8)

        img = cv2.imdecode(npimg, cv2.IMREAD_COLOR)

        img = cv2.resize(img, None, fx=0.4, fy=0.4)

        height, width, channels = img.shape

        blob_file = cv2.dnn.blobFromImage(img, 0.00392, (416, 416),
(0, 0, 0), True, crop=False)

        net.setInput(blob_file)
        out_point = net.forward(output_layers)
```

```
        boxes = []
        confidences = []
        class_ids = []

        for out in out_point:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]

                if confidence > 0.5:
                    center_x = int(detection[0] * width)
                    center_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)

                    point_x = int(center_x - (w / 2))
                    point_y = int(center_y - (h / 2))

                    boxes.append([point_x, point_y, w, h])
                    confidences.append(float(confidence))
                    class_ids.append(class_id)

        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

        for i in range(len(boxes)):
            if i in indexes:
                point_x, point_y, w, h = boxes[i]
                label = str(classes[class_ids[i]])

                result_query_recurso_turistico = []

                cur = mysql.connection.cursor()

                if request_type == 'seeObject':
                    cur.execute('''select * from trecursoturistico
where slug=%s and tipo=%s limit 1''',
                                (label, 'Reconocimiento de objeto',))

                if request_type == 'mapDraw':
                    cur.execute('''select * from trecursoturistico
where slug=%s and tipo=%s limit 1''',
                                (label, 'Reconocimiento de mapa',))

                result_query_recurso_turistico = cur.fetchall()

                payload = []
                content = {}

                for result in result_query_recurso_turistico:
                    content = {
                        'codigoRecursoTuristico': result[0],
                        'nombre': result[1],
                        'descripcion': result[2],
                        'ubicacion': result[3],
                        'latitud': result[4],
                        'longitud': result[5],
                        'clima': result[6],
                        'extensionImagen': result[7],
                        'extensionResenia': result[8],
                        'etiquetas': result[9],
```

```
                        'slug': result[10],
                        'iframe': result[11]
                    }

                    payload.append(content)
                    content = {}

                object_json.append({
                    "label": label,
                    "point_x": point_x,
                    "point_y": point_y,
                    "width": w,
                    "height": h,
                    "query_data": payload
                })

        return object_json

api.add_resource(Recognition, '/recognition')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port='5000', debug=True)
```

Anexo 3. Código JavaScript para la identificación de puntos de interés en Ionic

```
export class PointPage
{
    languaje: boolean;

    watchPosition: any;
    watchHeading: any;

    isStartCamera: boolean;
    cameraPreviewOpts: CameraPreviewOptions;
    widthScreen: number;
    heightScreen: number;

    arrayInterestPoint: Array<any>;

    myLatitude: number;
    myLongitude: number;

    iframe: string;
    isActiveShowDataInfo: boolean;

    constructor(
        private navController: NavController,
        private interestPointService: InterestPointService,
        private cameraPreview: CameraPreview,
        public toastController: ToastController,
        private geolocation: Geolocation,
        private deviceOrientation: DeviceOrientation,
        private platform: Platform,
        private storage: Storage
    )
    {
        this.isStartCamera=false;
        this.widthScreen=window.screen.width;
        this.heightScreen=window.screen.height;

        this.iframe=environmentVar.auxSrcIframe;
        this.isActiveShowDataInfo=false;

        this.cameraPreviewOpts=
        {
            x: 0,
            y: 0,
            width: this.widthScreen,
            height: this.heightScreen,
            camera: 'rear',
            tapPhoto: false,
            previewDrag: false,
            toBack: true,
            alpha: 0
        };

        this.myLatitude=null;
        this.myLongitude=null;

        this.arrayInterestPoint=[];
    }

    async ngOnInit()
```

```
    {
        this.languaje=(await this.storage.get('languaje'));

        this.interestPointService.JsonRest().subscribe((res: any) => {
            if(res.correcto)
            {
                this.arrayInterestPoint=res.listaTPuntoInteres;

                this.arrayInterestPoint.forEach((element: any) => {
                    element.showFullInfo=false;
                    element.side=null;
                });
            }
        });

        this.platform.ready().then(() => {
            this.getGeolocation();
        });

        this.showCamera();
    }

    getGeolocation()
    {
        this.watchPosition=this.geolocation.watchPosition({ maximumAge: In-
finity, timeout: Infinity, enableHighAccuracy: true }).subscribe((geoposi-
tion: Geoposition) => {
            this.myLatitude=geoposition.coords.latitude;
            this.myLongitude=geoposition.coords.longitude;
        });

        this.watchHeading=this.deviceOrientation.watchHeading().sub-
scribe((data: DeviceOrientationCompassHeading) => {
            let angleRespectNorth=data.magneticHeading;

            if(this.myLatitude!=null && this.myLongitude!=null)
            {
                this.arrayInterestPoint.forEach((element: any) =>
                {
                    element.distanceKm=this.calculateDistance(this.myLongi-
tude, element.longitud, this.myLatitude, element.latitud);

                    if(element.distanceKm>7)
                    {
                        return true;
                    }

                    let angleBetweenPoints: number=this.calculateAngleTwoPo-
sition(this.myLatitude, this.myLongitude, element.latitud, element.longitud);

                    let angleDiff=angleRespectNorth-angleBetweenPoints;

                    if(Math.abs(angleDiff)<=7 || (element.dis-
tanceKm==0 && Math.abs(angleDiff)<=15))
                    {
                        element.side='center';
                    }
                    else
                    {
```

```
                        element.side=((angleDiff<0 && angleDiff>-180) || (an-
gleDiff>=0 && angleDiff>180)) ? 'right' : 'left';
                    }
                });
            }
        });
    }

    calculateDistance(longitudeOne: number, longitudeTwo: number, lati-
tudeOne: number, latitudeTwo: number)
    {
        let p=0.017453292519943295;
        let temp=0.5-Math.cos((latitudeOne-latitudeTwo)*p)/2+Math.cos(lati-
tudeTwo*p)*Math.cos((latitudeOne)*p)*(1-Math.cos(((longitudeOne-longi-
tudeTwo)*p)))/2;
        let distance=(12742*Math.asin(Math.sqrt(temp)));

        return Math.trunc(distance);
    }

    toRadians(degrees: number)
    {
        return degrees*Math.PI/180;
    }

    toDegrees(radians: number)
    {
        return radians*180/Math.PI;
    }

    calculateAngleTwoPosition(startLat: number, startLng: num-
ber, destLat: number, destLng: number)
    {
        startLat=this.toRadians(startLat);
        startLng=this.toRadians(startLng);
        destLat=this.toRadians(destLat);
        destLng=this.toRadians(destLng);

        let y=Math.sin(destLng-startLng)*Math.cos(destLat);
        let x=Math.cos(startLat)*Math.sin(destLat)-
Math.sin(startLat)*Math.cos(destLat)*Math.cos(destLng-startLng);
        let brng=this.toDegrees(Math.atan2(y, x));

        return (brng + 360)%360;
    }

    showCamera()
    {
        this.isStartCamera=true;

        this.cameraPreview.setFlashMode('off');
        this.cameraPreview.setColorEffect('none');

        this.cameraPreview.startCamera(this.cameraPre-
viewOpts).then((res: any) => {},
        (err: any) => {
            this.presentToast(err);
        });
    }
```

```
    stopCamera()
    {
        if(this.isStartCamera)
        {
            this.cameraPreview.stopCamera();
            this.isStartCamera=false;
        }
    }

    async presentToast(text: string)
    {
        const toast=await this.toastController.create(
        {
            message: text,
            duration: 7000
        });

        toast.present();
    }

    clickShowFullInfo(element: any)
    {
        this.iframe=element.iframe;

        (document.querySelector('.sectionViewData') as HTMLEle-
ment).style.paddingLeft='7px';
        (document.querySelector('.sectionViewData') as HTMLEle-
ment).style.paddingRight='7px';
        (document.querySelector('.sectionViewData') as HTMLEle-
ment).style.top='3px';

        document.querySelectorAll('.itemMove').forEach((item: HTMLEle-
ment) => {
            item.style.display='none';
        });

        document.querySelectorAll('.sectionViewData > div').fo-
rEach((item: HTMLElement) => {
            item.style.paddingBottom='0px';
            item.style.paddingTop='0px';
        });

        element.showFullInfo=true;

        this.isActiveShowDataInfo=true;
    }

    clickHideFullInfo(element: any)
    {
        this.iframe=environmentVar.auxSrcIframe;

        (document.querySelector('.sectionViewData') as HTMLEle-
ment).style.paddingLeft='95px';
        (document.querySelector('.sectionViewData') as HTMLEle-
ment).style.paddingRight='95px';
        (document.querySelector('.sectionViewData') as HTMLEle-
ment).style.top='0px';
```

```
        document.querySelectorAll('.itemMove').forEach((item: HTMLEle-
ment) => {
            item.style.display='block';
        });

        document.querySelectorAll('.sectionViewData > div').fo-
rEach((item: HTMLElement) => {
            item.style.paddingBottom='3px';
            item.style.paddingTop='3px';
        });

        element.showFullInfo=false;

        this.isActiveShowDataInfo=false;
    }

    clickClose()
    {
        this.stopCamera();

        this.watchPosition.unsubscribe();
        this.watchHeading.unsubscribe();

        this.navController.navigateRoot(['home']);
    }
}
```