UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA EL RECONOCIMIENTO DE COVID 19 EN IMÁGENES DE RADIOGRAFÍA DE TÓRAX

QUEZADA SANMARTIN NIXON DOUGLAS
INGENIERO DE SISTEMAS

MACHALA
2020

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA EL RECONOCIMIENTO DE COVID 19 EN IMÁGENES DE RADIOGRAFÍA DE TÓRAX

QUEZADA SANMARTIN NIXON DOUGLAS
INGENIERO DE SISTEMAS

MACHALA
2020

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA EL RECONOCIMIENTO DE COVID 19 EN IMÁGENES DE RADIOGRAFÍA DE TÓRAX

QUEZADA SANMARTIN NIXON DOUGLAS
INGENIERO DE SISTEMAS

RIVAS ASANZA WILMER BRAULIO

MACHALA, 17 DE DICIEMBRE DE 2020

MACHALA
2020

TESIS NIXON QUEZADA

INFORME DE ORIGINALIDAD

8%	7%	0%	4%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Técnica de Machala Trabajo del estudiante	3%
2	firebase.google.com Fuente de Internet	1%
3	Submitted to Universidad Carlos III de Madrid Trabajo del estudiante	1%
4	es.slideshare.net Fuente de Internet	1%
5	developer.android.com Fuente de Internet	<1%
6	repositorio.konradlorenz.edu.co Fuente de Internet	<1%
7	www.clarin.com Fuente de Internet	<1%
8	ftp.riken.jp Fuente de Internet	<1%
9	www.jujuyaldia.com.ar Fuente de Internet	<1%

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

El que suscribe, QUEZADA SANMARTIN NIXON DOUGLAS, en calidad de autor del siguiente trabajo escrito titulado DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA EL RECONOCIMIENTO DE COVID 19 EN IMÁGENES DE RADIOGRAFÍA DE TÓRAX, otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.

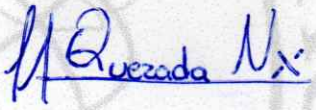
El autor declara que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

El autor como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 17 de diciembre de 2020



QUEZADA SANMARTIN NIXON DOUGLAS
0706432382

DEDICATORIA

El presente trabajo de titulación, se lo dedico de manera especial a mi madre y a toda mi familia, que siempre han sido el pilar fundamental en mi vida, con sus consejos y apoyo incondicional en todos los proyectos que me he trazado a lo largo de mi vida, porque sus palabras de motivación me han permitido superar cada obstáculo académico.

A mi hermano y sobrina, por ser esa fuente de inspiración, desde el inicio de mi carrera universitaria y a todas las personas que estuvieron pendientes de este proceso, que nunca dudaron de mis capacidades.

AGRADECIMIENTO

Agradezco de manera infinita a mi madre, quien desde el momento en que elegí esta carrera me ha apoyado y motivado en diferentes aspectos de esta etapa universitaria.

De manera especial quiero agradecer a las siguientes personas:

A mi tutor el Ing. Wilmer Rivas, quien con sus enseñanzas y constante asesoría hizo posible que el presente trabajo se desarrolle con éxito.

A los ingenieros Fausto Redrovan y Rodrigo Morocho, por compartir con sus alumnos los conocimientos, por la paciencia y dedicación, porque aparte de ser docentes, nos ofrecieron su amistad y tuvieron las palabras precisas en el momento necesario, por hacer de un salón de clases un lugar acogedor.

A la Ing. Jazmín Eras, con quien tuve el honor de compartir varios semestres de la carrera como compañera de clases y se convirtió en una gran amiga brindándome su apoyo desinteresado en todos los aspectos.

RESUMEN

En la actualidad, nos enfrentamos a un grave problema sanitario por la rápida propagación del virus covid-19 que no da tregua, a inicios de la pandemia, se pensaba que dicho virus solo atacaba a personas de edad avanzada, pero con el pasar de los meses y ante las tasas de contagios diarios y clasificados por edad nos permite entender la gravedad del caso y evidencia la inestabilidad y deficiencia del sistema sanitario de los países pero principalmente en Ecuador donde los recursos médicos no se dieron abasto para atender la alta demanda de pacientes contagiados diariamente y en donde a pesar de los esfuerzos y tomas de muestras diarias no se podía emitir resultados en menor tiempo y prescribir medicamento para iniciar el proceso de recuperación y evitar nuevos contagios. Es por esta y otras razones que a un grupo de estudiantes de la ciudad de Madrid les surge la idea de crear un modelo que reconoce una infección por coronavirus con hasta un 97% de precisión en radiografías, esta investigación y creación busca la manera de facilitar la emisión de diagnósticos a pacientes.

Es por ello que, motivados en la existencia de un modelo de red neuronal ya creada y del papel fundamental que las aplicaciones móviles han ocupado en el área de la medicina, surge la idea de procesar y entrenar una red neuronal basado en la investigación previa se procede a entrenar la red neuronal en la aplicación Firebase, tomando el data set de imágenes de radiografías de GitHub y desarrollo de una aplicación móvil que permita evidenciar el funcionamiento de la red entrenada y que esta sea de gran ayuda para el personal médico reduciendo tiempo de repuesta, costos de diagnóstico y lo más importante prescripciones médicas eficientes para el tratamiento de pacientes contagiados.

El dataset que se ha utilizado para el entrenamiento fue obtenido del repositorio de GitHub y está formado por un total de mil (1000) imágenes que comprende imágenes de radiografías de tórax de pulmones sanos y pulmones enfermos con neumonía ocasionada por el virus del Covid-19.

Para lograr el entrenamiento óptimo de los modelos se omitió el clásico proceso de utilizar equipos con grandes recursos o lenguaje Python, en su lugar se utilizó una de las herramientas que ofrece Google, específicamente AutoML Vision Edge en su versión gratuita, que es una plataforma de desarrollo multiplataforma para

desarrollar código y/o entrenar modelos de redes neuronales. Para el desarrollo de la aplicación móvil se utilizó el IDE Android Studio con lenguaje java. La aplicación móvil consta de una interfaz sencilla de una única ventana en la cual se puede capturar una foto o cargar una imagen desde la galería y esta evaluarla con el modelo entrenado y por ende evidenciar el funcionamiento óptimo del mismo en la identificación y/o clasificación de las radiografías que tienen indicios de contagios de covid y ayudar a los médicos a diagnosticar a sus pacientes y para terminar en trabajos futuros se pueden incluir mayor cantidad de imágenes de radiografías con pulmones en otros estados para así optimizar la precisión del entrenamiento de la red neuronal.

Palabras claves

Firebase, Radiografías, Codiv-19, Aplicación móvil, AutoML Vison Edge

ABSTRACT

Currently, we are facing a serious health problem due to the rapid spread of the covid-19 virus that does not give truce, at the beginning of the pandemic, it was thought that this virus only attacked elderly people, but with the passing of the months and given the daily infection rates classified by age, it allows us to understand the seriousness of the case and shows the instability and deficiency of the health system of the countries, but mainly in Ecuador where medical resources were not sufficient to meet the high demand for patients infected daily and where, despite efforts and daily sampling, results could not be issued in less time and prescribed to start the recovery process and prevent new infections. It is for this and other reasons that a group of students from the city of Madrid came up with the idea of creating a model that recognizes a coronavirus infection with up to 97% accuracy in X-rays, this research and creation seeks a way to facilitate issuing diagnoses to patients

That is why, motivated by the existence of a neural network model already created and the fundamental role that mobile applications have played in the area of medicine, the idea of processing and training a neural network based on previous research arises. proceeds to train the neural network in the Firebase application, taking the data set of X-ray images from GitHub and developing a mobile application

that allows to demonstrate the functioning of the trained network and that this is of great help for medical personnel reducing time of response, diagnostic costs and most importantly efficient medical prescriptions for the treatment of infected patients.

The dataset that has been used for the training was obtained from the GitHub repository and is made up of a total of one thousand (1000) images comprising images of chest X-rays of healthy lungs and diseased lungs with pneumonia caused by the Covid-19 virus. .

To achieve optimal training of the models, the classic process of using equipment with large resources or Python language was omitted, instead one of the tools offered by Google was used, specifically AutoML Vision Edge in its free version, which is a platform for multiplatform development to develop code and / or train neural network models. For the development of the mobile application, the Android Studio IDE with java language was used. The mobile application consists of a simple interface with a single window in which you can capture a photo or upload an image from the gallery and evaluate it with the trained model and therefore demonstrate its optimal performance in the identification and / or classification. of X-rays that have signs of COVID infection and help doctors diagnose their patients and to finish in future work can include more images of X-rays with lungs in other states to optimize the accuracy of the network training neuronal.

Keywords

Firebase, Radiographs, Codiv-19, Mobile App, AutoML Vison Edge

CONTENIDO

Índice de Ilustraciones

Índice de tablas

INTRODUCCIÓN

Los avances tecnológicos, en el campo de la medicina han modificado las técnicas tradicionales de detectar cualquier tipo de enfermedad,

Durante esta emergencia sanitaria, se ha observado un incremento considerable de nuevas tecnologías para el área de la salud e investigación, desde la creación de aplicaciones inteligentes que cumplen funciones desde la detección de síntomas de contagio hasta el diseño de nuevos patrones de investigación para una posible cura del coronavirus.

Es por ello que basado en una investigación previa y con el antecedente de un algoritmo de entrenamiento de una red neuronal que permite identificar neumonía por coronavirus mediante fotografías de radiografías de rayos x, las cuales fueron obtenidas de un repositorio de GitHub para realizar las debidas pruebas de entrenamiento de la red neuronal.

Para el entrenamiento de la red se utilizó Firebase el cual es una app multiplataforma que permite entrenar modelos de redes neuronales con Tensorflow.

El presente documento se encuentra estructurado de la siguiente manera:

Capítulo 1: en este capítulo se explica la utilidad en la implementación de una aplicación móvil que permita la detección de pacientes con covid-19 mediante imágenes de rayos x.

Capítulo 2: se detallan definiciones, fundamentación teórica de la aplicación desarrollada; así como objetivos, diseño, ejecución y pruebas de aplicación.

Capítulo 3: se establecen los resultados de la aplicación desarrollada, además de conclusiones y recomendaciones basadas en los objetivos propuestos al inicio del proyecto.

1. CAPÍTULO I. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS

1.1. Ámbito de Aplicación: descripción del contexto y hechos de interés

En la actualidad el mundo está enfrentando una pandemia, un virus llamado COVID 19, el cual está cobrando la vida de miles de personas alrededor del mundo, los métodos para detectar esta enfermedad se han desarrollado contra reloj debido al alto nivel de contagios, lo que evidencia la necesidad de tener opciones queden resultados en periodos cortos de tiempo, se requiere de igual forma pruebas que den resultados en cortos periodos de tiempo. Es por este motivo que se ha propuesto implementar una aplicación móvil que detecte COVID 19 en imágenes de radiografías de tórax para que el personal médico tenga en sus manos una herramienta que lo ayude en el proceso del diagnóstico de la enfermedad.

El reconocimiento de imágenes es un proceso que sigue una serie de pasos complejos y sucesivos en los cuales se convierte los datos de entrada en información que puede ser procesada por un ordenador, para posteriormente entrenar una red neuronal que facilita la toma de decisiones. El reconocimiento de imágenes con redes neuronales artificiales se ha utilizado en proyectos como la clasificación y mapeo automático de coberturas del suelo en imágenes satelitales [1] y en otros campos como el área de medicina para la Identificación del cilindro nudoso en imágenes de radiografía. [2]

El proceso de entrenar una red neuronal artificial se puede comparar en cierta forma al proceso de aprendizaje de un humano para identificar objetos, dicho proceso para un humano es utilizando el sentido de la vista y almacenando dicha información en su cerebro. En el caso de un ordenador no es para nada simple ya que este entiende solo en lenguaje binario y es aquí en donde se deben aplicar algoritmos y procesos que conviertan estas imágenes de datos que una computadora pueda procesar para entrenar una red neuronal y tomar decisiones en base a este entrenamiento.

Existen varias herramientas que permiten entrenar redes neuronales a partir de imágenes tales como Yolo, OpenCV, Café, entre otros. Además de aplicaciones web que facilitan este entrenamiento como Firebase de Google. La principal diferencia entre esta última aplicación es que el proceso de

entrenamiento no se ejecuta en un computador local, sino en un servidor remoto. Creando así modelos en cuestión de pocos minutos u horas sin una demanda excesiva de hardware.

La presente propuesta tecnológica tiene como objetivo desarrollar una aplicación móvil que permita detectar la presencia de coronavirus en imágenes de radiografía de tórax, con un banco de imágenes obtenido de internet, que sirvió para entrenar la red neuronal.

1.2. Establecimiento de requerimientos

Firebase Start UML, es una plataforma de desarrollo móvil que se encuentra en la nube de Google para diferentes plataformas móviles lo que representa una alternativa óptima de desarrollo de aplicaciones móviles.

El sistema de detección de covid en imágenes de radiografía de tórax se divide en las siguientes fases:

Recolección de información, en esta fase se obtuvo las imágenes de radiografías de tórax para entrenar la red, el data set que contiene las imágenes se encuentra en el repositorio de GitHub.

Entrenamiento del modelo con Firebase Auto ML intervienen dos tipos de radiografías de pacientes: de neumonía por coronavirus y la otra con ambos pulmones sanos, para entrenamientos posteriores se pueden utilizar otro tipo de radiografías para darle a la red un mayor grado de análisis y procesamiento de información.

Para el desarrollo app móvil, se utilizó Android Studio que es una herramienta ideal para desarrollar las interfaces necesarias para mostrar las respuestas del procesamiento de imágenes de la red neuronal entrenada.

La fase de pruebas se realizó después de completar el entrenamiento, con las imágenes procesadas, se procede a realizar las respectivas pruebas de detección de covid por medio de imágenes de radiografías de tórax.

1.3. Justificación de requerimientos a satisfacer

El presente proyecto tiene su enfoque en el dominio de investigación de Tecnologías que son aplicadas en los procesos productivos, que van ligados a

la línea de investigación Tecnologías para la producción sustentable establecido por la UTMACH.

En noviembre del año 2019, se registraba el primer contagio de COVID-19, la enfermedad causada por el virus SARS-COV-2, según investigación del periódico hongkonés South China Morning Post basada en datos gubernamentales. Como es de conocimiento general, dicho virus produjo una pandemia, creando caos en el sistema sanitario de los países subdesarrollados lo que provocó la muerte de miles de personas que no fueron diagnosticadas a tiempo y no recibieron la atención médica necesaria.

Para el desarrollo de esta aplicación, se ha considerado estudios recientes cuyos resultados detallan las características clínicas y paraclínicas de COVID-19, estos estudios han evidenciado los patrones característicos de rayos x de tórax y ha dado paso a la creación de algoritmos preentrenados para ser probados y/o modificados ajustándolos a las necesidades del mismo.

Firebase Star UML, es una plataforma de desarrollo para diferentes aplicaciones móviles, como en este caso que la aplicación permitirá ayudar al personal médico a dar respuestas en el menor tiempo posible, lo que evidenciaría optimización de recursos en tiempo y dinero.

2 CAPITULO II: DESARROLLO DEL PROTOTIPO

2.1 Definición del prototipo tecnológico

Las redes neuronales convolucionales a diferencia de las redes neuronales tienen un mayor porcentaje de eficiencia y por ende reducen los parámetros en la red, además de que permiten manejar imágenes de mejor calidad a gran escala. Firebase, se posiciona como una herramienta clave en el entrenamiento de la red neuronal, debido a que se puede entrenar a partir de fotografías de radiografías de tórax.

La arquitectura de la red neuronal convolucional consta de 2 fases: entrenamiento y predicción; la fase de entrenamiento inicia en la búsqueda, selección y extracción de imágenes necesarias, posteriormente las imágenes seleccionadas son subidas a un proyecto de Firebase, donde se debe elegir la latencia y tamaño del paquete del proyecto, para este caso se eligió la de uso general; cabe destacar que entre más óptimo sea el procesamiento más tiempo tomará.

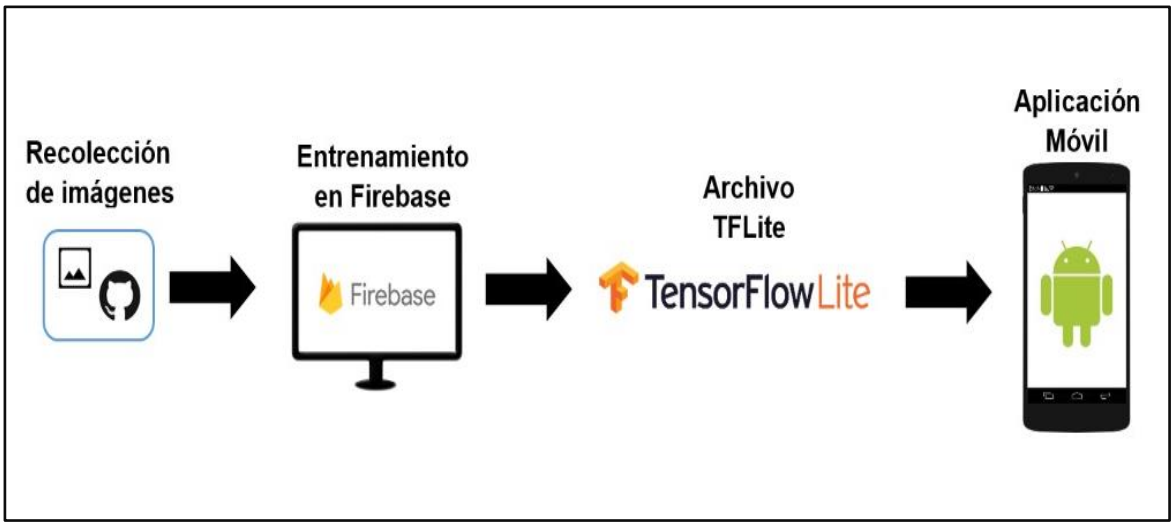


Ilustración 1 Arquitectura de entrenamiento Red Neuronal

Cuando el tiempo de entrenamiento se cumple, la aplicación notifica por medio de correo electrónico, se procede a descargar el archivo tflite que contiene los pesos y un txt con las etiquetas, estos archivos son los que posteriormente se utilizarán para desarrollar la aplicación en Android studio.

2.2 Fundamentación Teórica del Prototipo

2.2.1 Obtención de Imágenes

2.2.1.1 Imágenes médicas

Las imágenes médicas son el método más eficaz para el diagnóstico de enfermedades. Sin embargo, la interpretación manual requiere demasiado tiempo y esfuerzo porque este tipo de imágenes contienen un gran volumen de información. El diagnóstico asistido por computadora está desempeñando un papel cada vez más importante en la clínica, lo que puede ayudar los profesionales de la salud a analizar imágenes médicas automáticamente [3].

La red neuronal artificial es una técnica de clasificación atractiva fácil de implementar y se ha utilizado en una variedad de aplicaciones del mundo real [4]. Se han identificado varios estudios científicos actuales, en los que se ha utilizado redes neuronales para el análisis de imágenes médicas [5], y se ha probado la eficiencia en la detección de algunas patologías como cáncer de mama [6], enfermedades cardiacas [7] [8], y enfermedades pulmonares [9]; ya sea utilizando imágenes de tomografías [10], imágenes de rayos X [11] o imágenes de ultrasonidos [12]. Estos estudios se orientan a lograr una alta precisión de sus resultados, en algunos casos se ha realizado una segmentación para hacer la detección específicamente en el área problemática.

2.2.1.2 Github

"GitHub es el sitio de alojamiento de código fuente colaborativo más grande construido sobre el sistema de control de versiones de Git [13]." Es un software basado en web que se puede utilizar como medio para colaborar en el desarrollo de proyectos de software de aplicación, permitiendo la gestión de código fuente [14]. Es una plataforma que muchos desarrolladores usan no solo para controlar la versión de su proyecto, sino también para compartirlo [15].

GitHub se utiliza ampliamente para proyectos de investigación [16]. La disponibilidad de una API completa ha convertido a GitHub en un objetivo para muchos esfuerzos de investigación de colaboración en línea e ingeniería de software [13].

GitHub incluye herramientas de desarrollo como seguimiento de problemas (informes de errores y solicitudes de funciones), notificaciones, diferencias y paneles de estado; así como características sociales. Hay tres opciones para documentar un proyecto en GitHub: Readmes, wiki y GitHub Pages. Todas estas opciones se pueden crear en la sintaxis de Markdown. Debido a que los documentos se almacenan en el mismo repositorio que el código, pueden seguir el mismo flujo de trabajo, lo que es especialmente útil en un entorno ágil [17].

2.2.2 Construcción de la Red Neuronal

2.2.2.1 Firebase

La plataforma Firebase proporciona un gestor base de datos NoSQL basada en la nube que permite almacenar y sincronizar datos entre dispositivos en tiempo real [18]. Los datos actualizados entre nodos y procesados en milisegundos junto con los nodos móviles fáciles de usar brindan una excelente experiencia de usuario independientemente de la conectividad de la red [18].

Esta base de datos es ampliamente utilizada para almacenar datos de aplicaciones móviles que requieren de un almacenamiento en la nube, con el propósito de dar solución a diversos problemas; tales como la detección de enfermedades [19], minimizar el desperdicio de comida [20], gestión de membresías en gimnasios [21], verificación de la calidad del agua [22], entre muchas otras novedosas aplicaciones.

Las características clave de Firebase consisten en capacidad de procesamiento de datos en tiempo real y simplificación del desarrollo de aplicaciones. Estas características determinan a Firebase como el sistema gestor de base de datos más adecuado para aplicaciones móviles, ya que ofrece un mejor tiempo de respuesta [23].

Las funciones principales con las que inicio Firebase al unirse a Google en el 2016, fueron Realtime Database, User Authentication y Hosting, las cuales siguen facilitando el desarrollo de aplicaciones; pero la plataforma ha ido extendiéndose, por lo cual actualmente también cuenta con Firebase ML, una

herramienta que permite a los desarrolladores añadir capacidades de aprendizaje automático en sus aplicaciones móviles [24].

2.2.2.2 AutoML Vision Edge

El aprendizaje automático automatizado (AutoML) es una solución prometedora para construir un sistema de aprendizaje profundo sin ayuda humana y se está estudiando ampliamente [25].

AutoML Vision Edge se usa para entrenar un modelo con imágenes propias y sus respectivas categorías. El modelo personalizado se entrena en Google Cloud y, una vez que el modelo está listo, se usa por completo en el dispositivo [26].

Entre las aplicaciones más importantes de AutoML está la clasificación de imágenes biomédicas [27] que ayuda a profesionales de la salud facilitándoles la mejor utilización de modelos de aprendizaje automático en la detección precisa de enfermedades. Se ha comprobado que este tipo de técnicas automatizadas puede igualar o superar, en un período de tiempo más corto, el desempeño humano experto en ciertas tareas de aprendizaje automático [28].

2.2.3 Modelo Entrenado

2.2.3.1 TensorFlow Lite

Tesnorflow Lite, abreviado TF Lite, es una solución ligera que permite ejecutar modelos de aprendizaje profundo en dispositivos móviles e integrados. Si un modelo construido en TensorFlow o Keras puede convertirse con éxito al formato TensorFlow Lite, un nuevo formato de modelo basado en FlatBuffers, que es similar pero más rápido y mucho más pequeño en tamaño que ProtoBuffers, se puede esperar que el modelo se ejecute con una latencia baja y un tamaño binario más pequeño [29].

El flujo de trabajo básico de usar TensorFlow en sus aplicaciones móviles tanto para Android como para iOS se detalla en la siguiente ilustración, en donde los pasos comunes son construir el modelo inicial, entrenar el modelo TF, convertirlo usando TF Lite para obtener el archivo con el modelo; que en caso de ser utilizado para su implementación en una aplicación iOS, constará de una

API en C++ y un intérprete o kernel; mientras que si se utiliza para su implementación en una aplicación Android contará con una API en Java o C++, un intérprete y una Api de Red Neuronal [30].

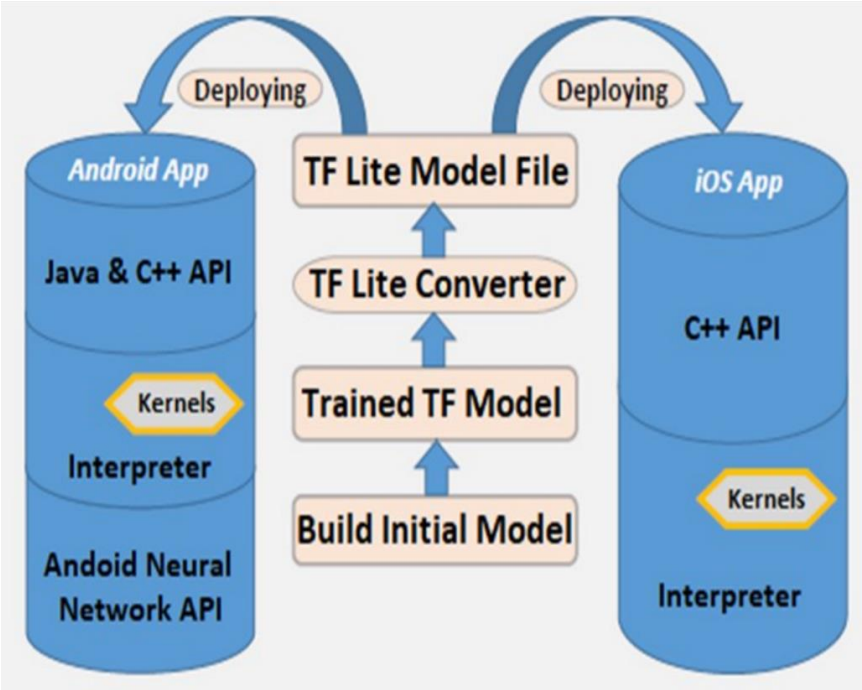


Ilustración 2. Diagrama de flujo de modelos de implementación de aprendizaje profundo en dispositivos móviles con TF Lite [30] .

"Existen investigaciones que hacen uso de aprendizaje profundo con TensorFlow Lite en smarthphones Android [31]." TensorFlow Lite permite a los usuarios ejecutar modelos de aprendizaje automático en dispositivos móviles con baja latencia, facilitando el reconocimiento de imágenes sin necesidad de consumir excesivamente recursos computacionales [30]. El funcionamiento básico consiste en convertir el modelo de archivo generado después de la fase de entrenamiento al archivo adecuado para el móvil con extensión .tflite [16].

2.2.3.2 Archivo txt

Un archivo que contiene caracteres organizados en cero o más líneas. Las líneas no contienen caracteres NUL y ninguna puede exceder los {LINE_MAX} bytes de longitud, incluido el carácter <newline>. Aunque POSIX.1-2017 no distingue entre archivos de texto y archivos binarios (consulte el estándar ISO C), muchas utilidades solo producen resultados predecibles o significativos

cuando operan con archivos de texto. Las utilidades estándar que tienen tales restricciones siempre especifican "archivos de texto" en sus secciones STDIN o INPUT FILES [32].

2.2.4 Construcción de Aplicación Móvil

2.2.4.1 Sistema Operativo Android

Android es un sistema operativo open source pensado para teléfonos móviles y desarrollado por la Open Handset Alliance (OHA) bajo autorización de Google. La OHA se compone de alrededor de 80 empresas, tales como Samsung, HTC, SFR, Orange, Asus, Qualcomm, entre otras [33].

Android se basa en un Kernel de Linux y se distribuye bajo licencia Apache License 2.0 [33]. Además de ser un sistema gratuito y multiplataforma, ha permitido instalarse en dispositivos móviles aun con gamas bajas. Esta plataforma móvil creada por la empresa Google se encuentra en el mercado desde septiembre del 2008 [34].

De acuerdo con el sitio web oficial: “Android está disponible para todo el mundo: desarrolladores, diseñadores y fabricantes de dispositivos, lo que significa que más personas tienen la posibilidad de experimentar, imaginar y crear cosas nunca antes vistas [35].”

Algunas características importantes de este sistema operativo se describen en la siguiente ilustración:

Ilustración 3. Características del Sistema Operativo Android [34]

2.2.4.2 Android Lollipop

Periódicamente se han desarrollado varias versiones del sistema operativo Android, las cuales generalmente incluyen mejoras para adaptarse a los requerimientos de los nuevos dispositivos móviles.

En el 2014 Google buscando más seguridad en su sistema operativo para dispositivos móviles lanzó Android Lollipop o Android versión 5.0, su logo o identificación gráfica era una chupeta, su última versión estable fue la 5.1.1 [36].

Esta versión se preocupa por el cifrado de la información de manera automática, cifrando todo el dispositivo en su primer arranque, también permite la configuración de las notificaciones que se desean ver con el dispositivo bloqueado, así como también teniendo en cuenta que en caso de pérdida del dispositivo existen opciones que permiten rastrearlo o borrar su contenido de manera remota, con el inconveniente que si llegasen a restaurarlo de fábrica, estas herramientas también perderían su utilidad y por ende con ella la opción de recuperar el aparato, por ello en esta versión se implementó la elección de contraseña para la ejecución de dicha restauración [36].

Las mejoras de seguridad que en su momento fueron introducidas en Android Lollipop impiden que se ejecuten la mayoría de las herramientas de adquisición

de memoria [37]. Una desventaja en este sistema operativo es que “en Lollipop, tenía que permitir todos los permisos de la aplicación antes de descargar la aplicación de Play Store [38].”

Android desde su versión 5.0, es capaz de interpretar y ejecutar directamente instrucciones expresadas en código binario Java (Java bytecode), antes de esta versión Android utilizaba una máquina virtual Dalvik con compilación en tiempo de ejecución, esto cambia a partir de la versión 5.0 en la que Android introduce el entorno Android Runtime, con el que compila el bytecode de Java en el proceso de instalación de la aplicación [39].

2.2.4.3 Android Studio

Android Studio es el IDE oficial para el desarrollo de software de la plataforma Android e incluye todo lo necesario para compilar aplicaciones del mencionado sistema operativo. Este IDE ha sido creado exclusivamente con el fin de acelerar el desarrollo y compilación de apps de la más alta calidad [40] y proporciona las herramientas eficaces para el programar aplicaciones que se adapten a los diferentes dispositivos Android [41]. Android Studio también se conoce como un IDE que requiere requisitos de computadora confiables por tener características sofisticadas y complejas [42].

Android Studio es un rico ecosistema de herramientas, incluidos Git y Gradle. Git, para el control de código fuente, y Gradle, una herramienta de prueba y compilación [43].

Cuenta con un sistema de compilación sólido y flexible que ofrece automatización de compilaciones, administración de dependencias y configuraciones de compilación personalizables. Permite la integración de Firebase y Cloud. Además, de admitir la incorporación de bibliotecas locales y alojadas [41].

2.2.4.4 Lenguaje Java

Al principio el lenguaje Java no se utilizaba mucho, pero con el rápido desarrollo de la World Wide Web e Internet, el lenguaje JAVA se ha convertido en un lenguaje popular [44].

Actualmente, Java forma parte de los lenguajes de programación más populares, y muchas aplicaciones y sitios web modernos se programan utilizando Java [45]. En este lenguaje se utilizan herramientas para el proceso de compilación como, JDK, JRE e IDE. Entre los entornos de programación o IDE para desarrollar en JAVA se encuentran Eclipse y NetBeans [46].

“El lenguaje más utilizado para la programación de aplicaciones Android es Java, por ser el más cercano a su entorno y máquina virtual [39].”

El lenguaje Java es excelente en funciones multiplataforma [47], es uno de los lenguajes de programación más utilizados para desarrollar muchas aplicaciones y sistemas [48], siendo la alternativa tradicional al momento de desarrollar una aplicación móvil.

Las aplicaciones de Android están escritas en Java, pero a diferencia de las aplicaciones de Java, tienen recursos limitados en cuanto a capacidad de almacenamiento y duración de la batería [49].

2.2.4.5 ML Kit de Aprendizaje Automático

ML Kit de Aprendizaje Automático (ML Kit AA) ofrece la experiencia en aprendizaje automático de Google a los desarrolladores de dispositivos móviles en un paquete potente y fácil de usar. Haga que sus aplicaciones de iOS y Android sean más atractivas, personalizadas y útiles con soluciones que están optimizadas para ejecutarse en el dispositivo [50].

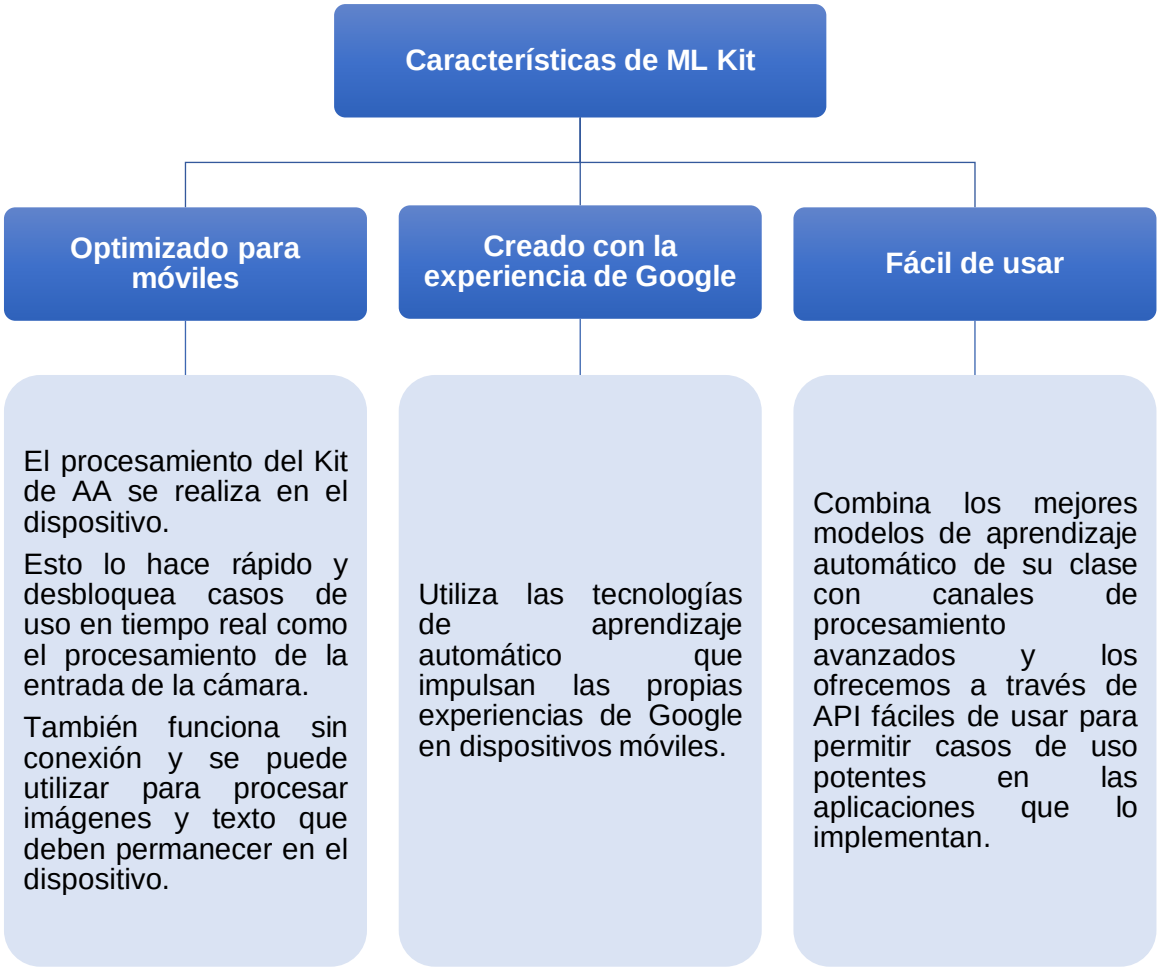


Ilustración 4. Características de ML Kit [50]

Las características que presenta ML Kit, hacen que sea una herramienta eficaz para su utilización en el desarrollo de una aplicación móvil, para el reconocimiento de Covid-19 en imágenes de radiografía de tórax.

2.2.4.5.1 API de Vision

El API de Vision es una herramienta que permite el análisis de video e imágenes para etiquetar imágenes y detectar códigos de barras, texto, caras y objetos [50].

En el siguiente gráfico se presentan las principales funcionalidades que tiene esta API de Google ML Kit de Aprendizaje Automático.

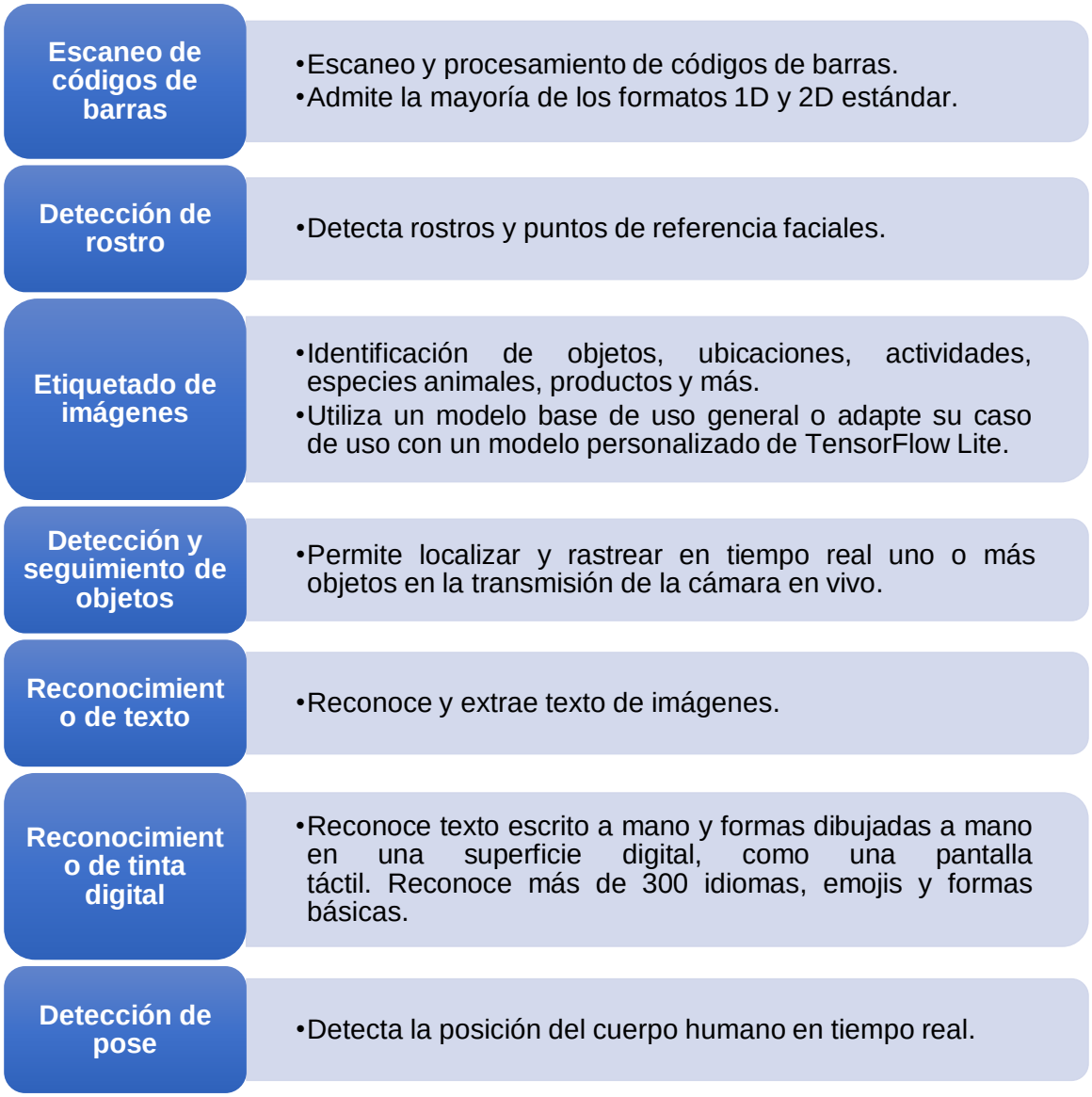


Ilustración 5. Funciones de la API de Vision [50]

2.2.4.5.2 Etiquetado de imágenes

Con las API de etiquetado de imágenes del Kit de AA, es posible reconocer entidades en una imagen sin tener que proporcionar metadatos contextuales adicionales. El etiquetado de imágenes brinda información valiosa sobre el contenido de las imágenes. Al usar la API, se obtiene una lista de las entidades que se reconocieron: personas, objetos, lugares, actividades y más. Cada etiqueta encontrada está acompañada de una puntuación que indica la confianza que tiene el modelo de AA en su relevancia. Con esta información, se

puede realizar tareas como la generación automática de metadatos y la moderación de contenido [51].

Específicamente en el desarrollo de la aplicación móvil, para el reconocimiento de Covid-19 en imágenes de radiografía de tórax, se utilizará ML Kit Vision label,

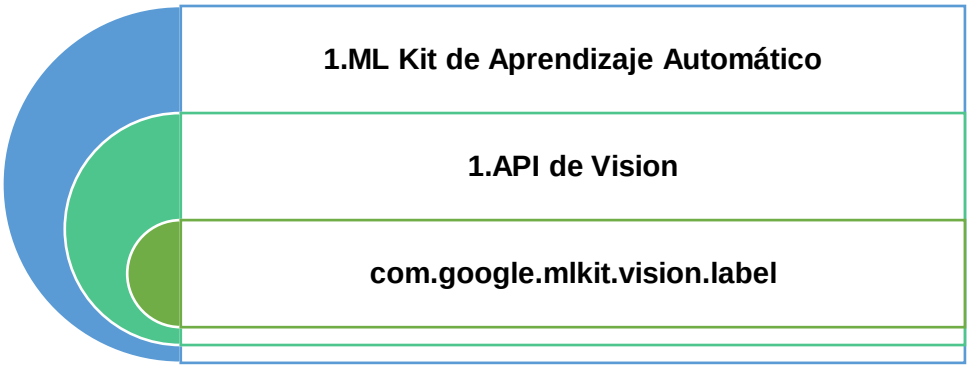


Ilustración 6. Ubicación de ML Kit Vision label [50]

ML Kit Vision label cuenta con la interfaz ImageLabeler, la cual le permite encontrar etiquetas en una imagen proporcionada. Además, dispone de tres clases, las mismas que se describen en la siguiente tabla [52].

ImageLabel	Representa una etiqueta de imagen detectada por ImageLabeler.
ImageLabelerOptionsBase	Opciones básicas para ImageLabeler.
ImageLabeling	Punto de entrada para obtener un valor ImageLabeler para encontrar ImageLabels en una imagen proporcionada.

Tabla 1. Clases de ML Vision label [37]

2.2 Objetivos del Prototipo

2.3.1. Objetivo General

Desarrollar una aplicación móvil que permita detectar la presencia de coronavirus, mediante imágenes de radiografía de tórax.

2.3.1. Objetivos Específicos

- Fundamentar científicamente los aspectos relacionados al entrenamiento de redes neuronales.
- Entrenar una red neuronal mediante un conjunto de datos obtenidos de un repositorio, para ser evaluadas por la aplicación.
- Ejecutar pruebas de entrenamiento y validación con la red neuronal entrenada.
- Desarrollar una aplicación móvil que permita realizar las pruebas de funcionamiento de la red neuronal.

2.3 Diseño del prototipo

El diseño del prototipo se divide en dos fases, primero el entrenamiento de una red neuronal convolucional que comprende desde la recolección de información y creación de la base de datos con las imágenes necesarias. Luego todas estas imágenes son subidas a un proyecto de Firebase, luego se escoge la Latencia y tamaño del paquete que deseamos que tenga nuestro proyecto y la otra fase la obtención del archivo que contiene los datos para utilizarlo en el desarrollo de la aplicación móvil, a continuación, se detallan los pasos del proceso de diseño:

1. Crear el proyecto en Firebase

Ilustración 7 Creación del proyecto en Firebase

2. Luego ir la opción de Machine learning y escoger AutoML Vision Edge

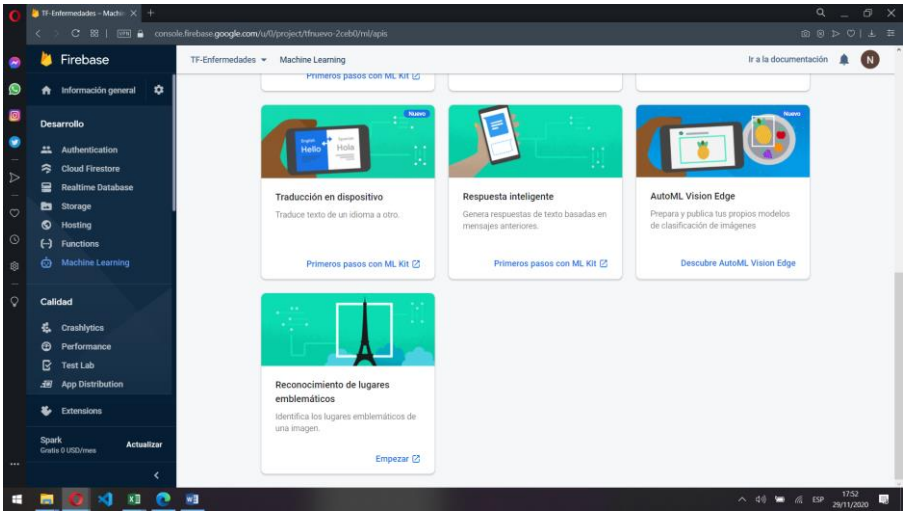


Ilustración 8 Herramientas de Machine Learning de Firebase

3. Crear un nuevo conjunto de datos

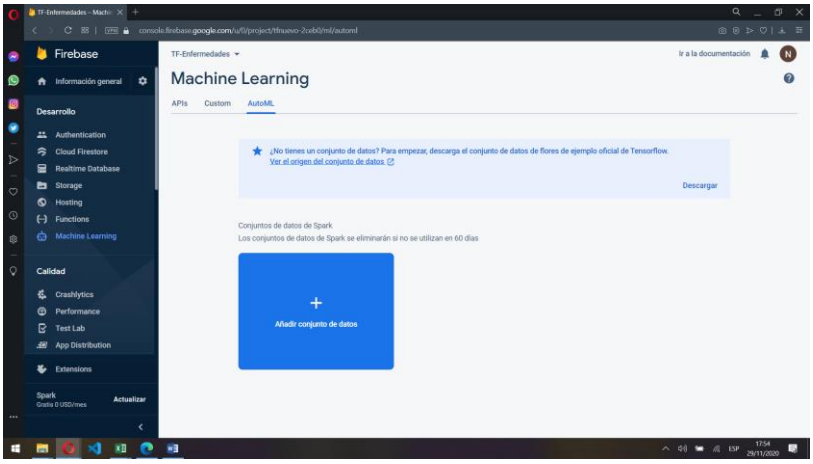


Ilustración 9 Ventana principal de la creación de proyecto con AutoML

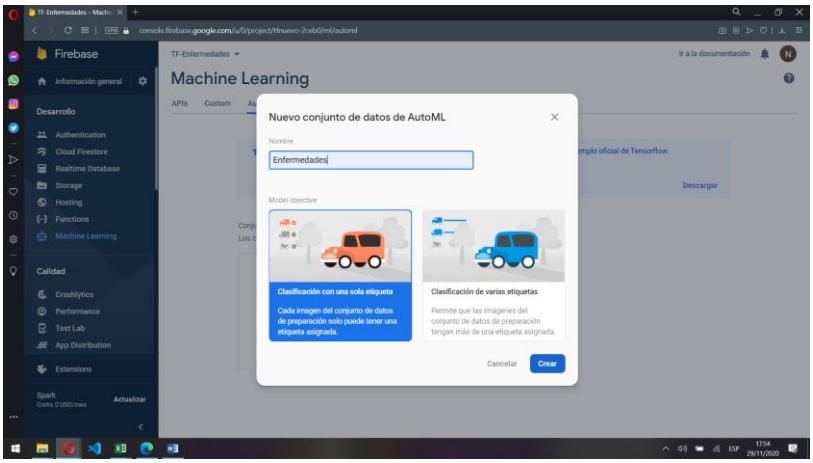


Ilustración 10 Creación del dataset

4. Subir el data set con las imágenes.

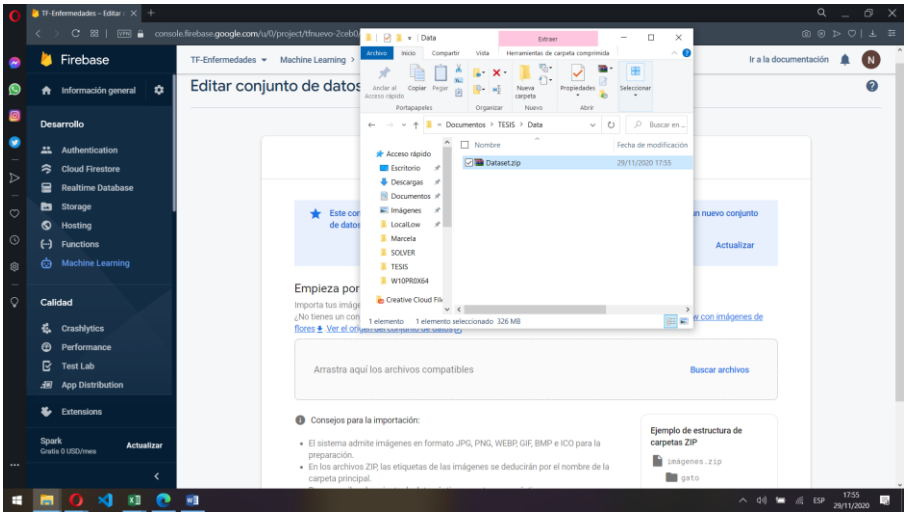


Ilustración 11 Importación del dataset local a Firebase

5. Una vez que se suben todas las imágenes etiquetadas son comprimidas según el nombre de la carpeta.

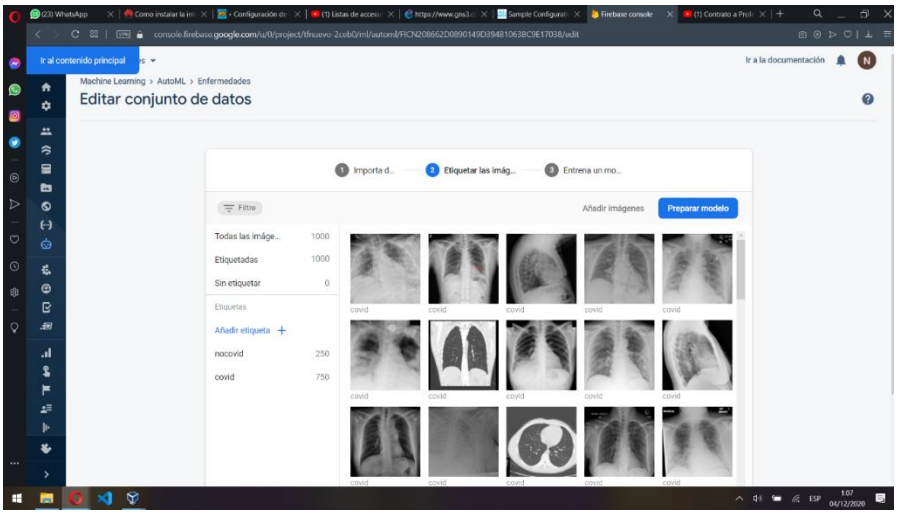


Ilustración 12 Conjunto de datos

6. Luego de subir, se asignan los parámetros que se desee que el modelo entrene.

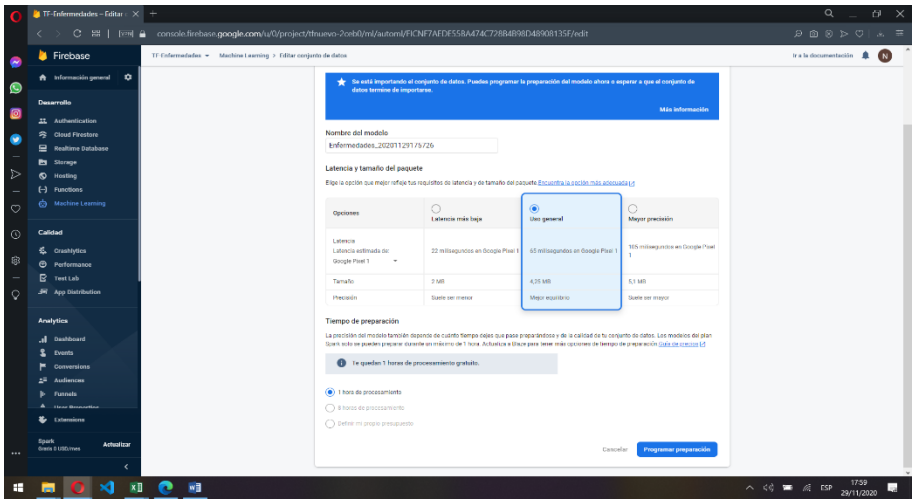


Ilustración 13 Preparación del modelo

7. Se debe programar la preparación y se espera un tiempo prudente hasta que llega un correo a la cuenta de registro, indicando que está listo, luego se puede descargar los archivos con los pesos que posteriormente se utilizarán para la aplicación móvil.

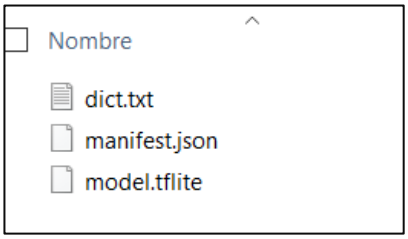


Ilustración 14 Archivos con los pesos del modelo

8. Maquetear la app

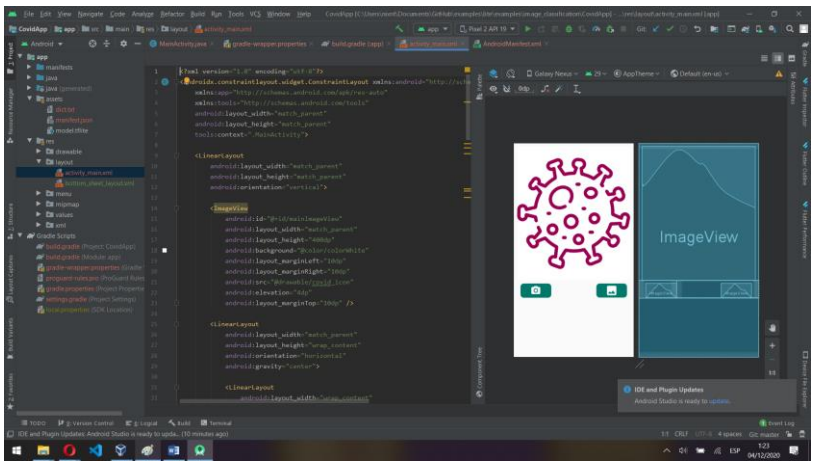


Ilustración 15 Diseño de la ventana principal

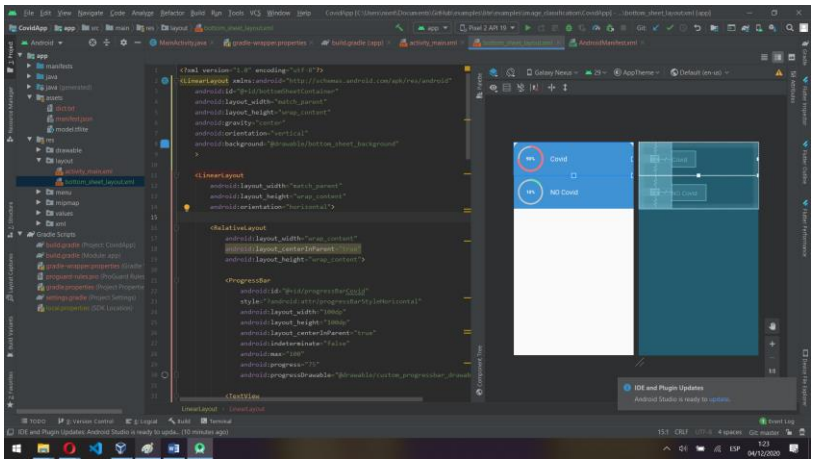


Ilustración 16 Diseño del mensaje de información

9. Digitar el código que interpreta la imagen con el modelo.

```
private void analyzeTfModel(Bitmap bitmap){
 AutoMLImageLabelerLocalModel localModel =
        new AutoMLImageLabelerLocalModel.Builder()
                .setAssetFilePath("manifest.json")
                .build();

 AutoMLImageLabelerOptions autoMLImageLabelerOptions =
        new AutoMLImageLabelerOptions.Builder(localModel)
                .setConfidenceThreshold(0.0f)  // Evaluate your model in the Firebase console
                // to determine an appropriate value.
                .build();
 ImageLabeler labeler = ImageLabeling.getClient(autoMLImageLabelerOptions);

 InputImage image = null;
  image = InputImage.fromBitmap(bitmap, 0);

 labeler.process(image)
        .addOnSuccessListener(new OnSuccessListener<List<ImageLabel>>() {
          @Override
          public void onSuccess(List<ImageLabel> labels) {
            for (ImageLabel label : labels) {
              String text = label.getText();
              float confidence = label.getConfidence();
              if(text.equals("covid")){
                dataConfidence[0] = confidence;
              }else {
                dataConfidence[1] = confidence;
              }
            }
            //show BottomSheet
            callBottomSheet(dataConfidence);
          }
        })
        .addOnFailureListener(new OnFailureListener() {
          @Override
          public void onFailure(@NonNull Exception e) {
            Toast.makeText(MainActivity.this, "Error al leer el modelo", Toast.LENGTH_LONG).show();
            Log.println(Log.ERROR, "ERROR:", "error en la lectura del modelo");
          }
        });
}
```

Ilustración 17 Código para interpretar el modelo

10. Realizar las debidas animaciones, botones y validaciones necesarias para la aplicación móvil.

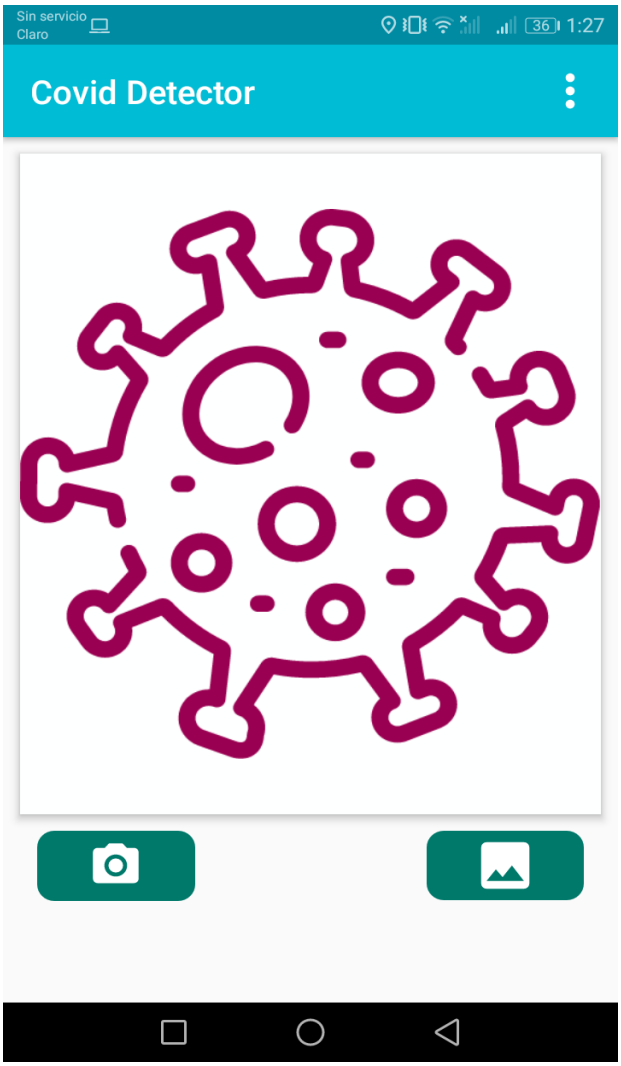


Ilustración 18 Diseño final de la app

2.4 Ejecución y/o ensamblaje del prototipo

2.4.1 Ejecución de la aplicación móvil

Descargar proyecto

Como primer se debe descargar la aplicación desde el repositorio de Github el cual se encuentra en el siguiente link https://github.com/nx96/CovidAppTesis

En Windows

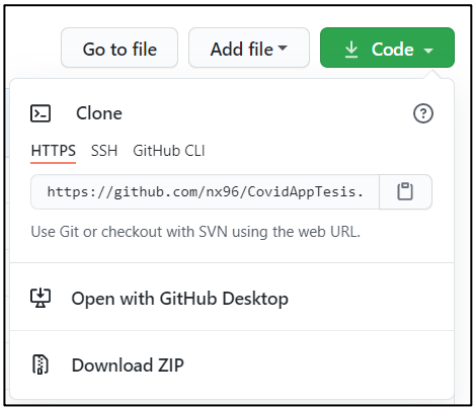


Ilustración 19 Descarga del repositorio

Para usuarios de Windows se puede descargar el repositorio desde el navegador o utilizando Github Desktop

En Mac

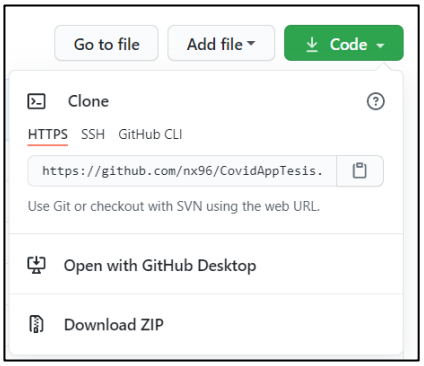


Ilustración 20 Descarga del repositorio

Los usuarios de Mac, al igual que los usuarios de Windows, pueden hacer uso de Github Desktop o también lo pueden descargar desde la terminal. Para ello deben abrir un terminal con la ubicación en la que desean tener el repositorio y ejecutar "git clone https:// https://github.com/nx96/CovidAppTesis"

En Linux

En Linux la manera más óptima para clonar un repositorio es ejecutando el comando "git clone https://github.com/nx96/CovidAppTesis" en la ubicación donde se va a guardar el repositorio.

Para ejecutar la aplicación hacemos uso del programa Android Studio

Para ejecutar Android Studio en Windows se debe acceder a su página oficial https://developer.android.com/studio, descargar el programa e instalarlo.

Dependiendo del sistema operativo que tengamos estos son los requisitos mínimos a tener en cuenta:

En Windows

- Es necesario una versión de Microsoft® Windows® 7/8/10 (64-bit)
- Al menos 4 GB de memoria RAM pero se recomienda 8 GB
- Tener 2 GB de espacio disponible en el disco duro, pero se recomienda 4 GB
- Y una resolución de pantalla mínima de 1280 x 800

En MAC

- Sistema operativo Mac OS X 10.10 (Yosemite) o superior
- El resto de requisitos son similares a los descritos anteriormente en Windows

En Linux

- El entorno de escritorio GNOME o KDE
- Una distribución de 64 bits que pueda ejecutar aplicaciones de 32 bits
- Biblioteca GNU C (glibc) 2.19 o superior
- El resto de requisitos son similares a los descritos anteriormente en Windows

Una vez descargado el proyecto se procede a abrir con el IDE Android Studio, para ello nos dirigimos a la barra de menú y escogemos la opción abrir

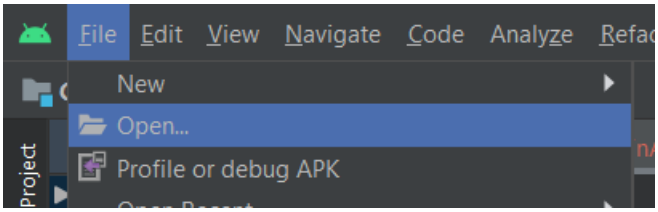


Ilustración 21 Menú abrir

Se abrirá un menú de navegación en donde buscaremos la locación del proyecto

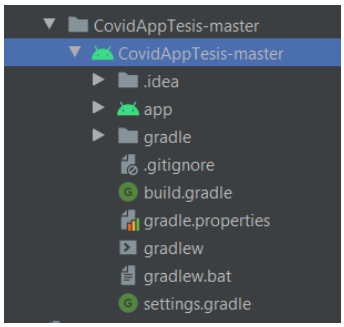


Ilustración 22 Estructura del proyecto

Luego procedemos a actualizar todas las librerías y dependencias necesarias escogiendo la opción **Make Project** dentro del menú **Build.** Este proceso puede tomar de uno a dos minutos.

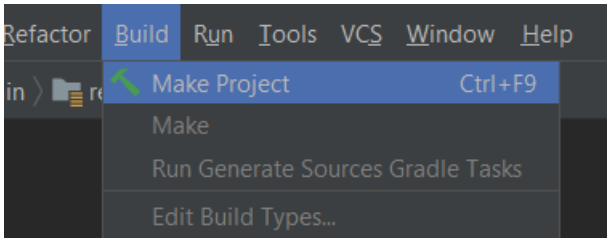


Ilustración 23 Opción Make Project

Además del proyecto en el ordenador, para ejecutar la aplicación móvil es necesaria un dispositivo físico (celular o tablet) con sistema operativo Android el cual se conecta al computador y por medio del cable de datos se procede a su ejecución

Una vez tengamos el proyecto listo solo falta ejecutar la aplicación móvil en un dispositivo físico o un emulador.

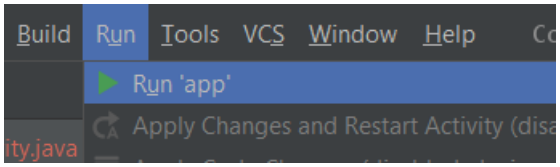


Ilustración 24 Opción Run 'app'

3. CAPITULO III: EVALUACIÓN DEL PROTOTIPO

3.1 PLAN DE EVALUACIÓN

El proceso de evaluación consta de dos partes, la primera es la evaluación del modelo para ello se utilizó métricas de clasificación. En el caso de la evaluación de la aplicación móvil al ser una aplicación de una única pantalla su rendimiento dependerá de la funcionalidad del modelo, por medio de librerías para su interpretación.

3.1.1 Prueba de Validación del modelo

3.1.1.1 Métricas de Clasificación

El resultado de la clasificación de imágenes con redes neuronales es un resultado binario, es decir la imagen pertenece a una clase predominante. En la ilustración 25 se explica la matriz de confusión, cuyos resultados son Verdaderos Positivos en caso de detectarse las clases de manera correcta, Falsos Positivos son los objetos que no pertenecen a las clases usadas en el entrenamiento, Falsos Negativos son los objetos que no se han detectado con el modelo y los Verdaderos Negativos son aquellos donde no se detecta un objeto que si fue entrenado. [53]

		Predicción de la Red Neuronal	
		Objeto Presente	Objeto no Presente
Situación Real	Objeto Presente	Verdadero Positivo	Falso Negativo
	Objeto no Presente	Falso Positivo	Verdadero Negativo

Ilustración 25Matriz de confusión Fuente [53]

Con estos datos se pueden calcular las métricas para la detección de objetos:

Exactitud

Se refiere a la cercanía del resultado con respecto a una prueba del valor real. Se calcula dividiendo la suma de verdadero positivo y negativo entre la suma de verdadero positivo, verdadero negativo, falso positivo y falso negativo. [54]

$$Exactitud = \frac{VP + VN}{VP + FP + FN + VN}$$

Precisión

Esta métrica determina que los objetos evaluados son acertados en la predicción. Para calcular este parámetro se divide el número de Verdaderos positivos entre la suma de Verdaderos positivos y falso positivo. [53]

$$Precisión = \frac{VP}{VP + FP}$$

Sensibilidad

Se obtiene al dividir el total de los verdaderos positivos entre total de verdaderos positivos y falso negativo. [53]

$$Sensibilidad = \frac{VP}{VP + FN}$$

Valor de referencia

Se obtiene dividiendo la multiplicación del valor obtenido en precisión por la sensibilidad entre la suma de los valores mencionados anteriormente. [53]

$$Valor\ de\ referencia = \frac{Precisión * Sensibilidad}{Precisión + Sensibilidad}$$

3.1.2 Prueba de evaluación de aplicación móvil

Para evaluar los resultados se desarrolló una aplicación móvil que utiliza librerías de java para leer el modelo entrenado, el archivo tflite, y mediante pruebas con imágenes ajenas a las utilizadas de manera inicial, para el entrenamiento, de pulmones sanos y pulmones enfermos de Covid19 se evidencio el resultado del proyecto.

3.2 RESULTADOS DE LA EVALUACIÓN

3.2.1 Resultados de validación del modelo

Tabla 2 Parámetros de evaluación

Parámetros	Valor
Conjunto de datos	Github – Mendeley Data
Imágenes de entrenamiento	1000
Imágenes de prueba	100
Latencia	105ms
Precisión	Alta

Tabla 3Resultados de evaluación

Métricas	Valor
Exactitud	0.98
Precisión	0.96
Sensibilidad	1.00
Valor de referencia	0.49

Los resultados de la evaluación indican que el rendimiento del modelo es más que satisfactorio. Esto debido a que entre las opciones para el entrenamiento del modelo con Firebase se utilizó la precisión alta, que aunque el tiempo de respuesta tiene un ligero retardo los resultados son los esperados.

3.2.2 Resultados de evaluación de la aplicación móvil

Resultado de la Prueba de Radiografías de Pulmones Enfermos

En las siguientes imágenes se evidencia que, una vez procesada la imagen por la red neuronal entrenada, nos da como respuesta que las siguientes imágenes tienen indicios de neumonía por covid-19, cumpliendo de esta manera con el propósito de identificar las radiografías que presentan dicha enfermedad.

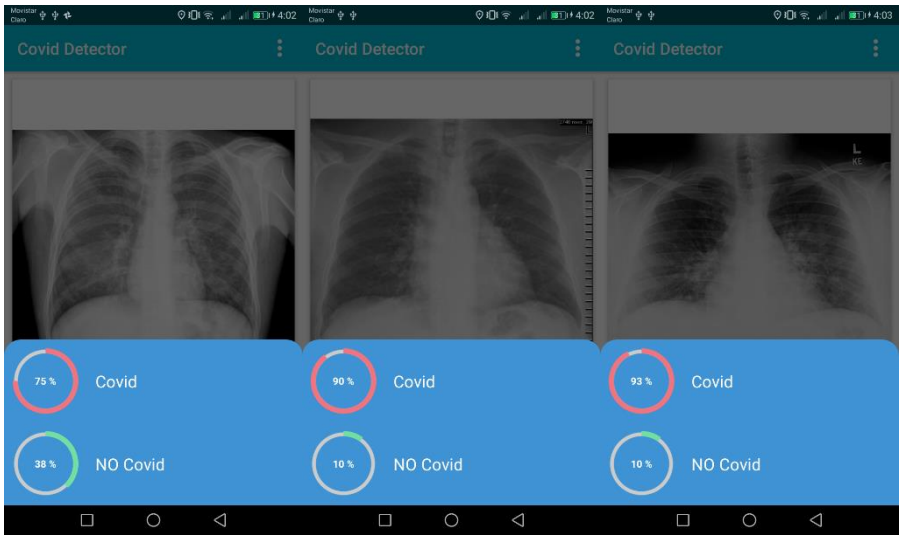


Ilustración 26 Pruebas pulmones enfermos

Resultados de imágenes de radiografías de pulmones sanos

En este caso las imágenes procesadas evidencian pulmones sanos, es decir que el modelo de la red neuronal no evidencia posible presencia de neumonía por covid-19.

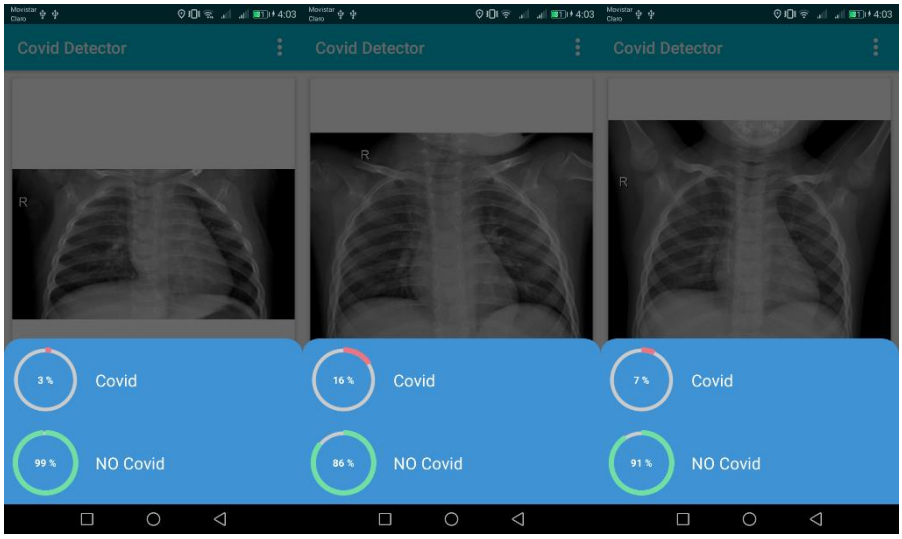


Ilustración 27 Pruebas pulmones sanos

3.3 CONCLUSIONES

- El uso de repositorios públicos validados facilita la obtención de imágenes para el entrenamiento ya que se conforman por aportes de una gran cantidad de personas y evita perder mucho tiempo en la recolección de datos para la investigación.
- Al realizar las pruebas de entrenamiento se validad que el modelo entrenado sea optimo y cumpla con el mínimo indispensable requerido. Estas pruebas son claves ya que al no realizarlas se puede obtener resultados de falsos positivos, quedando así el proyecto obsoleto.
- Desarrollar una aplicación móvil rápida y fácil de usar permite que al usuario no le tome mucho tiempo la capacitación para su utilización y reducir de manera considerable el tiempo de respuesta para emitir un diagnostico en un paciente.

3.4 RECOMENDACIONES

- Se recomienda mejora la base de datos de imágenes con archivos de una buena resolución, clasificados correctamente y solo de fuentes confiables, como pueden ser repositorios validados por expertos en el área de la medicina.
- Para futuros mejoras en el proyecto se recomienda aumentar el número de imágenes para el entrenamiento y de esta manera obtener un modelo más óptimo y preciso.
- Realizar más pruebas comparando entre las herramientas tradicionales para el entrenamiento de redes neuronales como lenguaje Python, Yolo, etc.
- Para futuras investigaciones, sería de gran aporte que se agregue nuevos módulos a la aplicación de modo que permita llevar un control personalizado de pacientes, analizar varias imágenes de manera simultánea y aumentar las clases de entrenamiento para detectar nuevas enfermedades.
- Se recomienda desarrollar la misma aplicación móvil para el sistema operativo iOS y de esta forma ampliar el número de usuarios para las pruebas de validación del modelo entrenado.

Bibliografía

[1] A. F., M. Castro-Franco y A. Cruz-Roa, «Clasificación y mapeo automático de coberturas del suelo en imágenes satelitales utilizando Redes Neuronales Convolucionales,» *Orinoquia,* vol. 21, nº 1, pp. 64-75, 2017.

[2] J. Karsulovic, M. Dinator, J. Morales, V. Gaete y B. Andres, «IDENTIFICACION DEL CILINDRO NUDOSO EN IMÁGENES TC DE TROZAS PODADAS DE PINUS RADIATA UTILIZANDO REDES NEURONALES ARTIFICIALES,» *Maderas: Cienca y tegnologia,* vol. 12, nº 3, pp. 229-240, 2010.

[3] S. Lu, S.-H. Wang y Y.-D. Zhang, «Detecting pathological brain via ResNet and randomized neural networks,» *Heliyon,* vol. 6, nº 12, 2020.

[4] A. Khan, J. Bukhari, J. I. Bangash, A. Khan, M. Imran, M. Asim, M. Ishaq y A. Khan, «Optimizing connection weights of functional link neural network using APSO algorithm for medical data classification,» *Journal of King Saud University - Computer and Information Sciences,* 2020.

[5] H. R. Boveiri, R. Khayami, R. Javidan y A. Mehdizadeh, «Medical image registration using deep neural networks: A comprehensive review,» *Computers & Electrical Engineering,* vol. 87, 2020.

[6] M. Desai y M. Shah, «An anatomization on Breast Cancer Detection and Diagnosis employing Multi-layer Perceptron Neural Network (MLP) and Convolutional Neural Network (CNN),» *Clinical eHealth,* 2020.

[7] Y. You, L. Viktorovich, J. Qiu, K. Nikolaevich y B. Y. Vladimirovich, «Cardiac magnetic resonance image diagnosis of hypertrophic obstructive cardiomyopathy based on a double-branch neural network,» *Computer Methods and Programs in Biomedicine,* 2020.

[8] D. Liu, Z. Jia, M. Jin, Q. Liu, Z. Liao, J. Zhong, H. Ye y G. Chen, «Cardiac magnetic resonance image segmentation based on convolutional neural network,» *Computer Methods and Programs in Biomedicine,* vol. 197, 2020.

[9] S. Hassantabar, M. Ahmadi y A. Sharifi, «Diagnosis and detection of infected tissue of COVID-19 patients based on lung x-ray image using convolutional neural network approaches,» *Chaos, Solitons & Fractals,* vol. 140, 2020.

[10] R. Yamauchi, S. Tamaki, Y. Morizane, S. Kusaka, Y. Manabe, Y. Akiyama, F. Sato y I. Murata, «Feasibility study on image reconstruction for single-photon emission computed tomography with limited projections by neural networks,» *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* vol. 986, 2021.

[11] S. Varela-Santos y P. Melin, «A new modular neural network approach with fuzzy response integration for lung disease classification based on multiple objective feature optimization in chest X-ray images,» *Expert Systems with Applications,* vol. 168, 2021.

[12] J. Su, Y. Liu y J. Wang, «Ultrasound image assisted diagnosis of hydronephrosis based on CNN neural network,» *Journal of King Saud University - Science,* vol. 32, nº 6, pp. 2682-2687, 2020.

[13] G. Gousios y D. Spinellis, «Mining Software Engineering Data from GitHub,» de *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, 2017.

[14] A. Zakiah y M. N. Fauzan, «Collaborative Learning Model of Software Engineering using Github for informatics student,» de *2016 4th International Conference on Cyber and IT Service Management*, Bandung, 2016.

[15] M. K. J., S. Dubey, B. Balaji, D. Rao y D. Rao, «Data Visualization on GitHub Repository Parameters Using Elastic Search and Kibana,» de *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, 2018.

[16] N. Kobayakawa y K. Yoshida, «How GitHub Contributing.md Contributes to Contributors,» de *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Turin, 2017.

[17] N. Bleiel, «Collaborating in GitHub,» de *2016 IEEE International Professional Communication Conference (IPCC)*, 2016 IEEE International Professional Communication Conference (IPCC), 2016.

[18] A. Darweesh, A. Abouelfarag y R. Kadry, «Real Time Adaptive Approach for Image Processing Using Mobile Nodes,» de *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Barcelona, 2018.

[19] İ. Güler, N. Baştürk, N. Samutoğlu y K. Küçük, «Real-Time Abnormal Detection for Asthma Patients with Internet of Things Technology,» de *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, 2018.

[20] H. Hajjdiab, A. Anzer, H. A. Tabaza y W. Ahmed, «A Food Wastage Reduction Mobile Application,» de *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Barcelona, 2018.

[21] D. Ward y C. Peoples, «An iOS Application With Firebase for Gym Membership Management,» *IEEE Potentials,* vol. 38, nº 3, pp. 27 - 34, 2019.

[22] T. Boonlar, «Sistema de verificación en línea para la calidad del agua potable de la máquina expendedora de agua potable,» *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS),* 2019.

[23] M. Ohyver, J. V. Moniaga, I. Sungkawa, B. E. Subagyo y I. A. Chandra, «The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test,» *Procedia Computer Science,* vol. 157, pp. 396-405, 2019.

[24] Firebase, «Firebase Machine Learning,» [En línea]. Available: https://firebase.google.com/products/ml. [Último acceso: 1 Diciembre 2020].

[25] X. He, K. Zhao y X. Chu, «AutoML: A survey of the state-of-the-art,» *Knowledge-Based Systems,* 2020.

[26] Firebase, «AutoML Vision Edge,» [En línea]. Available: https://firebase.google.com/docs/ml/automl-image-labeling. [Último acceso: 1 Diciembre 2020].

[27] A. Inés, C. Domínguez, J. Heras, E. Mata y V. Pascual, «Biomedical image classification made easier thanks to transfer and semi-supervised learning,» *Computer Methods and Programs in Biomedicine,* vol. 198, 2020.

[28] J. Waring, C. Lindvall y R. Umeton, «Automated machine learning: Review of the state-of-the-art and opportunities for healthcare,» *Artificial Intelligence in Medicine,* vol. 104, 2020.

[29] J. Tang, Intelligent Mobile Projects with TensorFlow: Build 10+ Artificial Intelligence apps using TensorFlow Mobile and Lite for iOS, Android, and Raspberry Pi, Birmingham: Packt Publishing, 2018, p. 404.

[30] A. Zeroual, M. Derdour, M. Amroune y A. Bentahar, «Using a Fine-Tuning Method for a Deep Authentication in Mobile Cloud Computing Based on Tensorflow Lite Framework,» de *2019 International Conference on Networking and Advanced Systems (ICNAS)*, Annaba, 2019.

[31] W. Campoverde, «Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con smartphones Android y Tensorflow Lite,» Dspace, Cuenca, 2019.

[32] IEEE and The Open Group, «The Open Group Base Specifications Issue 7,» 2018. [En línea]. Available: https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html#tag_03_403. [Último acceso: 1 Diciembre 2020].

[33] N. Benbourahla, Android 5: Principios del desarrollo de aplicaciones Java, ENI, 2015.

[34] Y. Sanguino, C. Romero y J. Jaramillo, «Aplicación de políticas de seguridad basadas en el estándar ISO 17799 para el control de riesgos personales del uso de la información en Smartphone con sistema operativo Android Versión Lollipop (5.0),» Repositorio Institucional UCC, Bucaramanga, 2017.

[35] Android Developers, «Android,» [En línea]. Available: https://www.android.com/intl/es_es/what-is-android/. [Último acceso: 1 Diciembre 2020].

[36] L. Cardenas, «SEGURIDAD EN DISPOSITIVOS MOVILES CON SISTEMA OPERATIVO,» Bogotá, 2020.

[37] H. Yang, J. Zhuge, H. Liu y W. Liu, «A Tool for Volatile Memory Acquisition from Android Devices,» de *IFIP International Conference on Digital Forensics*, 2016.

[38] S. R. Pawar y M. Kulati, «Recent Versions on Android and survey on the versions Lollipop,» *International Research Journal of Engineering and Technology (IRJET),* vol. 5, nº 3, pp. 1186-1189, 2018.

[39] F. Sánchez, «Comparativa Kotlin y Java en desarrollo Android,» Málaga, 2019.

[40] Android developers, «Notas de la versión de Android Studio,» Agosto 2020. [En línea]. Available: https://developer.android.com/studio/releases. [Último acceso: 1 Diciembre 2020].

[41] Android developers, «Android Studio,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 1 Diciembre 2020].

[42] E. Wihidayat y D. Maryono, «Pengembangan aplikasi android menggunakan Integrated Development Environment (IDE) App Inventor 2,» *Jurnal Ilmiah Edutic,* vol. 4, nº 1, 2017.

[43] C. Craig y A. Gerber, Learn Android Studio: Build Android Apps Quickly and Effectively, Apress, 2015.

[44] Y. Jiang, «Research on Application Value of Computer Software Development in Java Programming Language,» *Journal of Physics: Conference Series,* vol. 1648, 2020.

[45] F. Shahjamali, An Empirical Analysis of Java Language Use in Open Source Applications, Kingston: Queen's Graduate Theses and Dissertations, 2019.

[46] C. Garzón, Introducción al lenguaje de programación Java, 2019.

[47] L. Guo, B.-f. Quian, G.-x. Lin y B.-r. Pan, «Design of Remote Real-Time Measuring System,» *Current Trends in Computer Science and Mechanical Automation ,* vol. 2, pp. 1-17, 2017.

[48] M. Abdallah y M. Ai-Rifaee, «Java Standards: A Comparative Study,» *International Journal of Computer Science and Software Engineering (IJCSSE),* vol. 6, nº 6, pp. 146-151, 2017.

[49] Y. Cheon y A. Escobar, «Impacts of Java Language Features on the Memory,» Digital Commons, El Paso, 2017.

[50] Google Developers, «ML Kit,» [En línea]. Available: https://developers.google.com/ml-kit. [Último acceso: 1 Diciembre 2020].

[51] Firebase, «Etiquetado de Imágenes,» [En línea]. Available: https://firebase.google.com/docs/ml-kit/label-images. [Último acceso: 1 Diciembre 2020].

[52] Google Developers, «com.google.mlkit.vision.label,» [En línea]. Available: https://developers.google.com/android/reference/com/google/mlkit/vision/label/package-summary#classes. [Último acceso: 1 Diciembre 2020].

[53] W. B. Rivas Asanza, F. K. Farias Rivera y J. A. Guerrero Guerrero, «Implementación de un sistema para el conteo volumétrico de objetos mediante redes neuronales convolucionales,» 2019. [En línea]. Available: http://repositorio.utmachala.edu.ec/handle/48000/13940.

[54] J. I. Barrios Arce, «La matriz de confusión y sus métricas,» 2019. [En línea]. Available: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/.