Universidad Europea

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN

ANÁLISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

Sistema anti-plagio: Detección del uso de aplicaciones de chat en tiempo real

NOMBRE:

PABLO ARCONES PUEBLA

CURSO 2021-2022

TÍTULO: Sistema anti-plagio: Detección del uso de aplicaciones de chat en tiempo real

AUTOR: PABLO ARCONES PUEBLA

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS (BIG DATA)

DIRECTOR DEL PROYECTO: RAÚL PÉRULA MARTÍNEZ

FECHA: Octubre de 2022

RESUMEN

Este trabajo analiza los casos de plagio y copia en trabajos y exámenes del entorno académico. En él se analiza y recogen las definiciones y consecuencias de su realización, así como los diferentes intervinientes del proceso y su responsabilidad.

Con el fin de comprender y prevenir estos actos inmorales, diferentes herramientas y tecnologías empleadas para la detección de los casos de copia han sido analizadas. De esta forma, este trabajo recoge aquellas más utilizadas en la actualidad y las conclusiones extraídas de su análisis.

Además, durante la realización de este proyecto se ha diseñado y desarrollado un sistema que permite detectar el uso de herramientas de chat mediante el procesamiento en tiempo real de capturas de pantalla de los estudiantes que realizan exámenes desde sus ordenadores personales, mostrando en este documento todo el proceso seguido para su desarrollo y las diferentes configuraciones probadas.

Palabras clave: Computer Vision, Machine Learning, Deep Learning, CNN

ABSTRACT

This thesis analyzes the cases of plagiarism and copying in papers and exams in the academic environment. It analyzes and collects the definitions and consequences of their realization, as well as the different participants in the process and their responsibility.

In order to understand and prevent these immoral acts, different tools and technologies used to detect cases of copying have been analyzed. In this way, this work gathers the most used ones at present and the conclusions drawn from their analysis.

In addition, during this project a system has been designed and developed to detect the use of chat tools by processing in real time screenshots of students taking exams from their personal computers, showing in this paper the whole process followed for its development and the different configurations tested.

Key words: Computer Vision, Machine Learning, Deep Learning, CNN.

AGRADECIMIENTOS

Agradezco a la Universidad Europea de Madrid la oportunidad de haber podido cursar este Master y los conocimientos que me ha proporcionado durante el curso.

También agradezco a mi padre, que me ha cuidado y acompañado durante toda mi vida, y que me ha enseñado los valores necesarios para no desistir nunca y lograr las metas que me propongo.

Universidad Europea

Índice

Índice de Figuras

Capítulo 1. INTRODUCCIÓN

1.1 Planteamiento del problema

Durante las últimas décadas, las personas de diferentes partes del mundo han sido espectadoras de numerosos avances tecnológicos que, a medida que pasaba el tiempo, se han integrado cada vez más en su día a día. Como resultado, un nuevo mundo interconectado ha estado creciendo, en el que sus integrantes disfrutan de un sinfín de herramientas que distorsionan los antiguos límites y reestablecen las formas de interactuar con muchos ámbitos de la vida.

Ámbitos como las relaciones interpersonales, antiguamente influenciadas por las grandes distancias o por limitaciones de tiempo, se han visto sumamente mejoradas con la aparición de redes sociales tales como Facebook, Twitter o WhatsApp entre otras. Estas aplicaciones han reestablecido la forma en la que las diferentes personas del mundo interactúan en su día a día con sus conocidos, e incluso han propiciado y facilitado entablar nuevas relaciones entre sus usuarios. Además, la rápida transmisión de información de este tipo de tecnologías ha convertido estas herramientas en excelentes plataformas de difusión.

Otro de los ámbitos que más se está viendo afectado con la aparición de las nuevas tecnologías es el académico. Con la aparición de nuevas herramientas también han surgido nuevas metodologías de enseñanza. Algunas de estas metodologías sustituyen el uso de libros y cuadernos por tabletas, ordenadores y equipos de virtualización con el fin de potenciar el aprendizaje de los alumnos y mejorar así la posición de las instituciones académicas en los rankings mundiales.

Sin embargo, los avances tecnológicos logrados hasta la fecha no solo han proporcionado beneficios a la sociedad, sino que también han supuesto la adaptación y aparición de diversas actividades y metodologías con fines no éticos.

La constante aparición de nuevas funcionalidades y herramientas supone un desafío que las entidades y personas deben analizar y afrontar con el fin de reducir el mal uso de la tecnología y evitar así las posibles repercusiones que estas puedan suponer para ellos.

Los diferentes centros formativos y entidades vinculadas a la educación han propuesto y utilizado, a lo largo de los últimos años, diferentes herramientas y metodologías con el fin de evitar que sus alumnos se aprovechen de las nuevas tecnologías para mejorar sus resultados en ejercicios y exámenes de forma ilícita. Sin embargo, a la hora de controlar el uso de redes sociales, muchas veces los docentes optan por confiscar de forma temporal o incluso controlar el uso de los dispositivos de forma presencial, suprimiendo así los beneficios de la tecnología y requiriendo de un esfuerzo excesivo

por parte del profesorado, haciendo que el uso de ordenadores y dispositivos móviles se convierta más en un problema que en una ventaja para la docencia.

Es por ello que en este trabajo se plantea la necesidad del uso de la tecnología para la creación de herramientas que permitan controlar el uso de redes sociales sin limitar con ello el uso de los dispositivos electrónicos, reduciendo el esfuerzo de los docentes y mejorando el aprendizaje de los alumnos.

1.2 Objetivo del proyecto

Con el crecimiento de las nuevas tecnológicas, cada vez más personas tienen acceso a las diferentes herramientas comentadas anteriormente. El número de usuarios ha llegado a suponer un problema a la hora de diseñar y desarrollar las herramientas de control que detectan y previenen actos inmorales en todo tipo de entornos. Además, nuevos escenarios como los surgidos durante la pandemia de la COVID-19 han hecho aparecer nuevos retos que previamente no se habían planteado.

Es por ello que surgen nuevos modelos de *big data* que permiten establecer controles y trabajar con grandes volúmenes de datos sin que el número de usuarios afecte de forma significativa a los resultados obtenidos.

El **objetivo principal** de este proyecto consiste en establecer un estudio sobre las diferentes tecnologías empleadas en el área académica para prevenir y detectar los casos de copia y plagio. Además, se desarrollará una herramienta que permita detectar el uso de herramientas de chat en tiempo real, utilizando tecnologías del ámbito *big data*, que permitan reducir los tiempos de detección incluso cuando el número de usuarios sea muy elevado.

Para ello se definen los siguientes **objetivos específicos** que han sido perseguidos durante la realización de todo el proyecto:

1. Analizar qué es la copia y el plagio y sus efectos sobre los diferentes integrantes del sector académico.
2. Recoger y analizar las diferentes herramientas y tecnologías empleadas en el sector académico para detectar el plagio y la copia.
3. Analizar las tecnologías *big data* que puedan suponer una mejora sobre los sistemas actualmente implementados o que permitan trabajar con grandes volúmenes de usuarios.
4. Diseñar y construir una herramienta que emplee tecnología *big data* para detectar el uso de herramientas de chat durante exámenes realizados en sistemas informáticos.

1.3 Estructura del proyecto

Este documento recoge los estudios y análisis desarrollados para la realización del proyecto, así como los diseños y especificaciones utilizados. De esta forma, los diferentes elementos se encontrarán distribuidos de la siguiente forma:

1. **Estado del arte**: En este apartado se encuentran los estudios realizados para comprender el significado del plagio y la copia en el entorno académico, además de recoger los diferentes estudios que determinan el estado del arte de las tecnologías empleadas para detectar y prevenir actos inmorales.
2. **Implementación de la solución**: En este apartado se recoge toda la información referente a cómo se ha realizado este proyecto. Entre sus apartados se podrán ver los recursos empleados, diseños y pasos seguidos.
3. **Pruebas y resultados**: En este apartado se encuentran los diferentes diseños y especificaciones llevadas a cabo para el desarrollo de la herramienta de detección. Se recogen los diferentes componentes y configuraciones probadas con el fin de detallar el procedimiento seguido y los resultados obtenidos.
4. **Conclusiones y Futuras líneas de trabajo**: En este apartado se muestran oportunidades de desarrollo e investigación que han surgido durante la realización del proyecto y que puedan suponer una mejora sobre los resultados o sobre el ámbito del proyecto

Capítulo 2. Estado del arte

2.1 Consecuencias de la copia y el plagio en el entorno académico y origen de su prevención

La creciente demanda de formaciones online y la aparición de nuevos centros formativos basados plenamente en cursos impartidos de forma remota han dado lugar a que las instituciones de formación presencial hayan centrado sus esfuerzos en este nuevo nicho. Para ello, han adaptado la metodología de aprendizaje y de evaluación de sus alumnos, de forma que estos puedan obtener sus titulaciones desde el lugar que mejor se adapte a sus necesidades.

Sin embargo, las formaciones online poseen ciertos factores limitantes a la hora de perseguir el plagio y la copia entre los alumnos. El estudiante, al no encontrase en un entorno vigilado por los docentes, encuentra una mayor facilidad y falta de controles para realizar esta actividad inmoral, esto, sumado a la influencia de otros alumnos puede llegar a convertirse en un aliciente.

El plagio es un elemento persistente que algunos estudiantes utilizan con la motivación de minimizar o reducir esfuerzos a la hora de aprobar una materia. Este acto, muchas veces es realizado sin que los estudiantes sean plenamente conscientes de las consecuencias que pueden acarrear para ellos mismos y para el resto de los integrantes del proceso formativo.

2.1.1 Consecuencias de la copia y el plagio y metodologías utilizadas

Las entidades que se pueden ver afectadas cada vez que un caso de copia o de plagio se da por parte de los alumnos son:

- **Centro formativo**: Si los casos de plagio no son controlados y son expuestos a los medios públicos, el centro puede ver afectada su reputación al mismo tiempo que puede perder posiciones en los diferentes rankings que conforman la base del prestigio del centro. De la misma forma, el centro puede perder patrocinios y oportunidades con terceros que no quieran verse afectados por la mala reputación de los centros, resultando todo ello en una posible pérdida económica significativa.
- **Docentes**: Si los casos de plagio son expuestos, los docentes pueden perder reputación en el mundo académico y, para aquellos que formen parte del mundo empresarial, también pueden hacerlo en las empresas en las que trabajen. Dando como resultado una pérdida de oportunidades laborales significativa.
- **Alumno**: Si un alumno es descubierto copiando o cometiendo plagio, se expone a medidas disciplinarias por parte del centro formativo que pueden llegar a ser tales como la expulsión del propio centro. Además, si los casos de plagio se hacen

públicos, es posible que las empresas rechacen la entrada del alumno a sus plantillas, dificultando el proceso de búsqueda de trabajo.

Con el fin de reducir ese riesgo reputacional y económico, los centros educativos a lo largo del mundo han desarrollado e implementado diferentes metodologías y herramientas que previenen la ocurrencia de estos casos. [1]

Entre estas metodologías se encuentran centros que han impartido en la totalidad del curso las materias de forma remota, sin embargo, debido a la falta de recursos y herramientas, las evaluaciones de las asignaturas se han realizado de forma presencial, evitando así la falta de controles. Por otra parte, muchos centros han basado sus controles en la realización de los exámenes al mismo tiempo que se realizaban videollamadas, para que, de esta forma, los docentes fuesen capaces de ver las caras y movimientos de los alumnos; sin embargo, esta metodología se encuentra limitada al no mostrar las aplicaciones que los alumnos tienen abiertas mientras realizan el examen.

Por otra parte, algunas entidades académicas han optado por el uso de herramientas tecnológicas como COLIBRI o INSPERA que permiten la realización de exámenes a través de plataformas o incluso el bloqueo de aplicaciones no deseadas durante el examen. Mientras, otras han optado por el uso de programas de '*proctoring*' (Monitorización de fotografías y reconocimiento facial, además del uso de información del uso del navegador) [2]. Este tipo de herramientas tecnológicas será el que se analizará y sobre el que se basará la investigación de esta tesis.

2.1.2 Proctoring y reconocimiento de imágenes

Las herramientas de proctoring son utilizadas con el fin de verificar la identidad de los alumnos y validar que su comportamiento durante el examen es licito. Para ello se basan en el reconocimiento facial a través de imágenes tomadas con la '*web cam*' del estudiante con el uso de modelos de inteligencia artificial.

De esta forma, las aplicaciones empleadas registran la aparición de más personas durante la realización del examen, el uso de otros dispositivos como móviles o *smartwatchs* o incluso el audio del sistema.

Sin embargo, existe cierta controversia entre quienes afirman que el uso de este tipo de herramientas reduce los casos de plagio durante la realización de exámenes online, y quienes defienden que el uso de estas herramientas no supone un cambio importante en la detección del plagio [3,4].

A continuación, se encuentran listadas algunas de las herramientas de proctoring que se pueden encontrar en el mercado y que facilitan controlar los exámenes online:

- Mercer – Mettl
- ProctorU
- Examity
- Verificient
- AIProctor
- Turnitin

2.2 Análisis de herramientas

De las herramientas descritas en el apartado anterior, se detallan a continuación aquellas que utilizan tecnología diferenciada y que tienen relación con este proyecto.

2.2.1 INSPERA

INSPERA es un software de pago que establece una plataforma en la que los docentes pueden crear exámenes interactivos digitales y generar procesos automatizados para ahorrar tiempo durante los mismos. Por otra parte, INSPERA proporciona a los docentes estadísticas sobre sus exámenes y facilita la comunicación entre docentes de todas partes del mundo.

Los controles que ofrece esta herramienta se dan durante y tras la realización de los exámenes. INSPERA dispone de dos elementos principales para la gestión de la copia:

- **Control de conexión de los alumnos**: Establece una pestaña en la que los docentes pueden ver si sus alumnos se encuentran conectados a la herramienta o no, y en el caso de no estarlo, el tiempo que ha estado reconectándose.

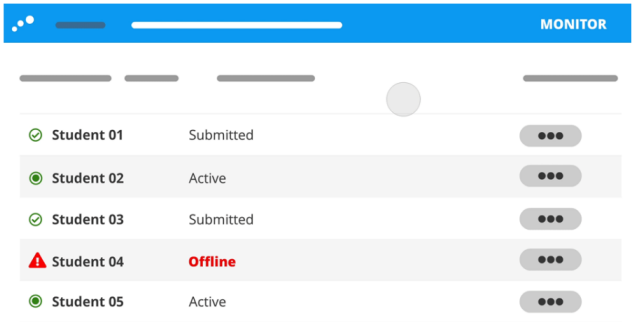


Ilustración 1: Inspera - Control de conexión

- **Generación de reportes automatizada en base a respuestas**: La herramienta analiza las respuestas de los alumnos y genera reportes en base a la similitud de las respuestas.

Gracias a estos controles, los docentes son capaces de conocer en todo momento el estado de sus alumnos durante el examen e identificar respuestas que sean similares entre ellos. [5]

2.2.2 Mercer – Mettl

Esta herramienta es una de las más avanzadas que utilizan la tecnología para controlar los exámenes, cursos y formaciones de los alumnos. En este caso, el uso del proctoring toma un papel fundamental, estableciendo los siguientes controles en sus plataformas [6]:

- Monitoreo de video en vivo y opciones de supervisión bajo demanda.
- Alertas automáticas por comportamiento sospechoso del usuario.
- Detección automática de rostro y ojos en intervalos regulares.
- Sesiones de revisión: grabación completa y revisión de toda la sesión.
- Supervisión offline a través de la cámara frontal para evaluaciones realizadas en tabletas / teléfonos.

Además, esta herramienta establece controles de autenticación para garantizar que sus alumnos son los mismos que realizan los exámenes:

- Restricciones de acceso por direcciones IP, ciertos dispositivos.
- Acceso bloqueado a reconocimiento facial
- Enlaces de prueba privados para evitar la distribución accidental.
- Autorización y control de forma remota.

Por último, también establecen controles de seguridad con el fin de garantizar la integridad de las pruebas y de los resultados utilizando los siguientes medios:

- Controles algorítmicos de vanguardia: prohibición de simuladores de cámara, uso compartido de pantalla, extensión de escritorio, persona extra.
- Control del navegador para evitar que los candidatos naveguen a otras páginas desde la ventana de prueba.
- Derechos de acceso definidos, para asegurar que no se manipulen las pruebas.
- Mantenimiento de informes de registro – registro de auditoría

2.2.3 AI Proctor

AI Proctor es una plataforma de proctoring desarrollada y mejorada con la ayuda de sistemas de inteligencia artificial. Gracias a ello, establece métodos de monitorización en tiempo real, controles sobre las búsquedas realizadas y el uso de aplicaciones no

utilizadas. Además, permite continuar con el uso de los controles incluso cuando no existe conexión a internet [7].

Algunos de los controles que esta herramienta implementa y que establece para controlar la copia son:

- Uso de la detección por voz y el reconocimiento facial para garantizar una experiencia similar a la de un examen presencial.
- Monitorización en tiempo real y generación de informes con las actividades sospechosas detectadas.
- Lista blanca de las páginas web y aplicaciones permitidas durante el examen.

2.3 Proyectos similares – Deep Learning

2.3.1 Natural Language Processing based Deep Learning

Uno de los desafíos más críticos a la hora de detectar el plagio y la copia documental es el análisis de la documentación. En este proceso, los docentes y sistemas dedican enormes cantidades de tiempo con el fin de detectar similitudes que puedan evidenciar un caso de plagio.

En el proyecto mostrado en el paper de la Universidad de Craiova titulado “NLP based Deep Learning Approach for Plagiarism Detection” emplean técnicas de Deep learning junto con procesamiento del lenguaje natural y modelos previamente entrenados para ofrecer un enfoque completamente nuevo a la detección del plagio.

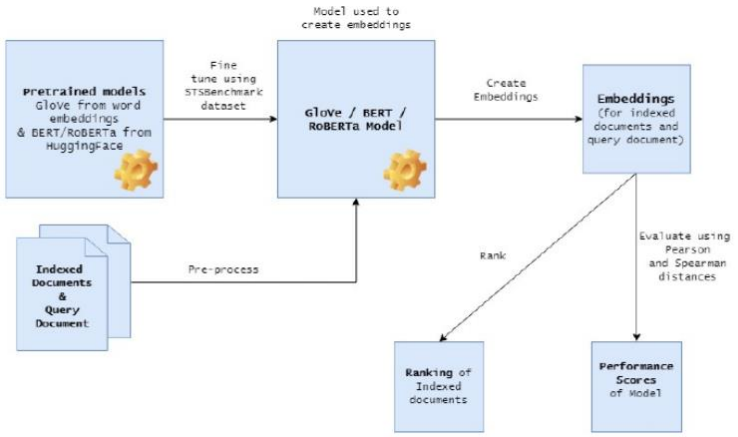


Ilustración 2: Flujo de analisis del proyecto NLP

Con el objetivo de evaluar su sistema y medir la efectividad con respecto a los sistemas anteriormente en uso, en este proyecto se realiza una comparativa entre los algoritmos de conteos de palabras y su nuevo sistema.

Como resultado de este proyecto ser obtuvo un sistema innovador que ofrece la capacidad de detectar casos de plagio incluso cuando varias técnicas han sido empleadas en el mismo documento.

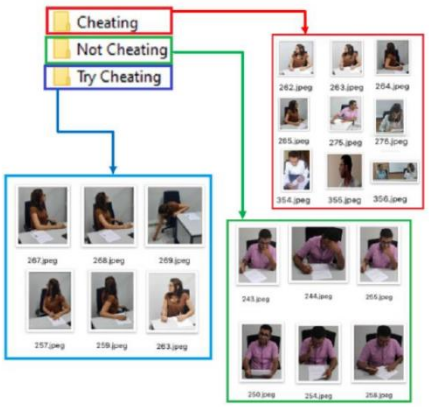
Sin embargo, al tratarse de un sistema nuevo, existen ciertos aspectos que requieren de mejoras como, por ejemplo, que el sistema ve reducida su tasa de acierto a medida que el documento es más largo [8].

Sin embargo, este proyecto permite ver como el uso de sistemas de Deep learning y de procesamiento del lenguaje mejoran los resultados obtenidos en sistemas convencionales y como su uso en este ámbito puede llegar a ofrecer mejoras significativas.

2.3.2 Convolutional Neural Network based system

Otro de los aspectos perseguidos durante la realización de exámenes y para la prevención de los casos de copia y plagio es el control sobre los movimientos extraños de los alumnos. Además, con la aparición de exámenes online en los que los examinadores no se encuentran presentes, la dificultad de detectar este tipo de casos se incrementa considerablemente.

En el paper "DEEP LEARNING: NEW APPROACH FOR DETECTING SCHOLAR EXAMS FRAUD" se encuentra definido un proyecto que utiliza la tecnología de Deep learning y de redes convolucionales con el objetivo de detectar este tipo de movimientos extraños que podrían indicar que un alumno estuviese copiando.



En este proyecto, a través de imágenes de los alumnos, tomadas con cámaras o web cams de los ordenadores de los estudiantes, desarrollan un modelo que trata de predecir entre si los alumnos están copiando, intentando hacerlo o si están haciendo el examen con normalidad.

Como resultado de este proyecto, se obtiene un modelo con una tasa de acierto sobre el data set de entrenamiento del 95% y un 89% sobre el data set de prueba, logrando

con ello un modelo plenamente funcional que permite reducir en gran medida los casos de copia y plagio en exámenes online y presenciales. Además, el modelo resultado es capaz de detectar los elementos situados alrededor del alumno y emplearlos en la predicción [9].

Con este proyecto se puede ver la gran efectividad lograda al emplear redes convolucionales en esta área y como su uso simplifica y reduce los costes de la realización de exámenes presenciales y online.

Capítulo 3. Implementación de la solución

3.1 Descripción de la solución

Tras el estudio realizado sobre las herramientas actualmente en uso, se ha detectado que en su gran mayoría establecen controles prohibitivos o métodos que requieren de una revisión intensiva por parte del profesorado para el control de la copia y el plagio en exámenes online.

Con el fin de solucionar este problema, en este apartado se expone la metodología seguida para el desarrollo de una nueva solución que detecta el uso de herramientas de chat en tiempo real de tal forma que el profesorado únicamente necesite revisar algunas pocas evidencias y no grabaciones completas de la sesión.

Este sistema consistirá en la toma de capturas de pantalla en los ordenadores en los que los alumnos realicen los exámenes, de tal forma que dichas capturas sean enviadas a un sistema de colas y posteriormente sean procesadas con el fin de detectar si los alumnos se encuentran utilizando una herramienta de chat o no.

Dado que el procesamiento de imágenes para detectar el uso de herramientas es muy complejo de realizar mediante una solución *ad hoc*, un data set de imágenes ha sido construido y empleado para entrenar una red neuronal que extraiga patrones de las capturas de escritorio y permita clasificar las nuevas imágenes en base a los suyos.

Con el objetivo de que el modelo empleado proporcione los mejores resultados posibles, un plan de pruebas ha sido desarrollado. En dicho plan de pruebas se han establecido las arquitecturas y meta-parámetros que serán testeados y evaluados para utilizar el que mejores resultados obtenga.

3.2 Arquitectura de la solución

A continuación, se muestran los recursos técnicos empleados para la realización de este proyecto.

3.2.1 Hardware

El proyecto objeto de esta memoria ha sido desarrollado utilizando un equipo MSI Raider GE76 con las siguientes características hardware:

- **CPU**: Procesador i7-12700H de 14 núcleos y 2.7 GHz
- **GPU**: Nvidia GeForce RTX 3070 Ti
- **Memoria RAM**: 32 GB DDR5
- **Almacenamiento**: SSO de 1 TB

Esta configuración ha facilitado en gran medida el desarrollo del proyecto, debido a la gran demanda de memoria y núcleos establecidos por el modelado con Keras y por el tamaño de las imágenes a procesar.

3.2.2 Software

A continuación, se muestra esquematizada la arquitectura técnica empleada para el proyecto. Dicha arquitectura recoge las diferentes tecnologías empleadas para el desarrollo del proyecto en las fases de modelado, entrenamiento y completo funcionamiento del sistema.

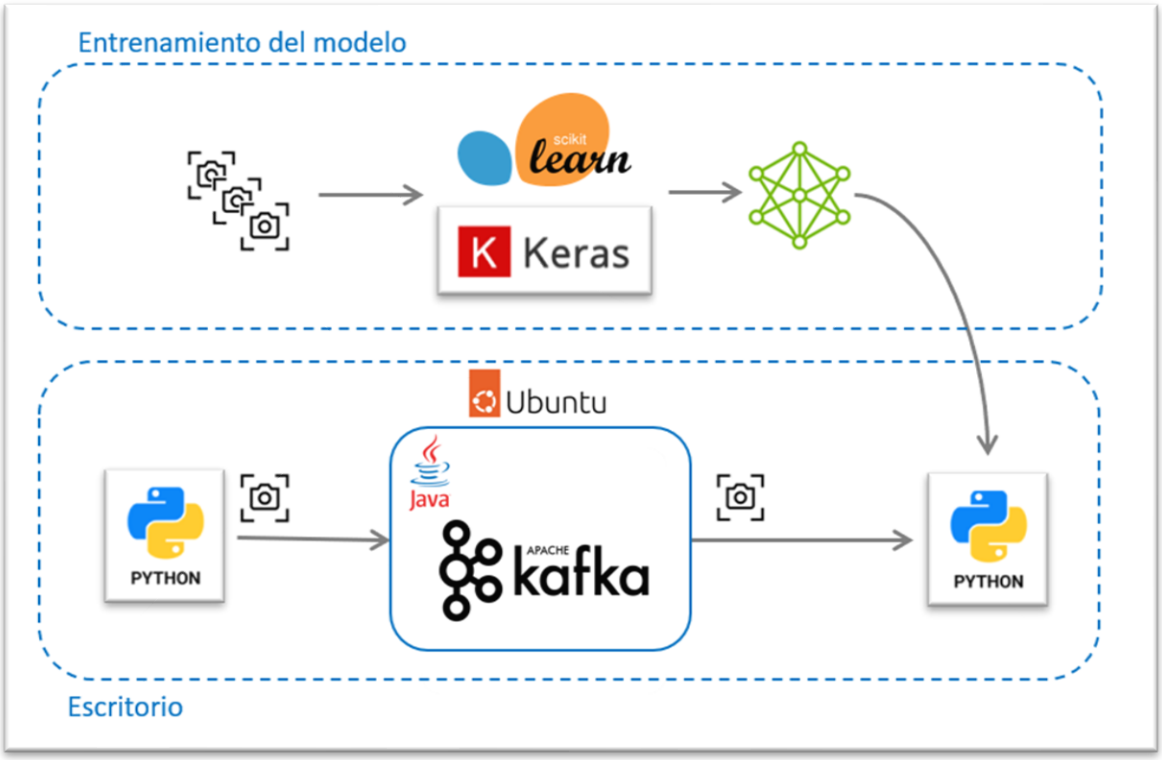


Ilustración 3: Arquitectura del sistema

La arquitectura del sistema ha sido construida e instalada sobre el sistema operativo Windows 11 Home, cuya compatibilidad con herramientas como VirtualBox, WhatsApp Desktop y la utilidad de captura de pantalla han facilitado el desarrollo del proyecto.

A continuación, se muestran resumidos los diferentes pasos que conforman el sistema, de acuerdo con el diagrama mostrado en la figura anterior:

1. Un modelo es construido y entrenado mediante el uso de Scikit-learn, Keras y un data set formado por imágenes de escritorios preprocesadas.
2. Uno o varios productores de Kafka desarrollados en Python obtienen y envían capturas de pantalla a una instancia de Apache Kafka.

3. La instancia de Apache Kafka instalada y configurada almacena y evita la perdida de imágenes realizadas por el productor, y facilita la paralelización de procesamiento en tiempo real.
4. Uno o varios consumidores de Kafka desarrollados en Python importan el modelo resultante del punto 1 y lo utilizan para predecir las diferentes imágenes recibidas de la instancia de Kafka. El número de consumidores será igual al número de particiones establecidas para el tópico de Kafka.

3.3 Planificación del proyecto

En este apartado se define la estructura, tecnologías y pasos seguidos para la gestión y desarrollo de este proyecto.

3.3.1 Gestión de versiones y repositorios

Los proyectos tecnológicos traen consigo una serie de riesgos inherentes que pueden afectar significativamente a su correcto desarrollo. Factores como apagones o fallos en los sistemas informáticos pueden suponer grandes pérdidas de información o incluso del total del proyecto.

Por otra parte, los diferentes avances y ajustes realizados durante el desarrollo de los proyectos suponen una constante aparición de nuevas versiones en documentos, archivos y otros componentes. Si estas versiones no son correctamente gestionadas, el acceso a la información o documentos puede llegar a suponer un incremento significativo del tiempo de desarrollo, suponiendo retrasos en las entregas.

Con el fin de prevenir estos dos problemas, en este proyecto se ha generado y utilizado un repositorio online en la plataforma GitHub, en el que a través de la herramienta GIT se ha gestionado la generación de versiones y su almacenamiento en la nube.

De esta forma, el uso de esta tecnología ha supuesto las siguientes ventajas:

- Generación de versiones ordenada.
- Acceso rápido a versiones anteriores de componentes del proyecto.
- Rápida visualización de cambios entre versiones.
- Respaldo del proyecto completo en la nube (Dificultando la perdida de avances)

3.3.2 Estructuración del proyecto

Una vez implementada la tecnología necesaria para la gestión de versiones del proyecto y antes de comenzar con la implementación del mismo, se ha establecido la estructura de carpetas con el fin de almacenar de forma ordenada los diferentes archivos y componentes del proyecto.

Dado que uno de los objetivos del proyecto es el desarrollo de un modelo de clasificación, se ha decidido utilizar una estructura que se basase en proyectos de *Data Science*. En concreto, el modelo tomado y adaptado para la realización de este proyecto es el ofrecido por **Cookiecutter**, que define una estructura estandarizada pero flexible a las diferentes necesidades de cada proyecto.

Partiendo de la estructura inicial proporcionada por la herramienta, se han realizado diferentes ajustes en los ficheros y estructuras de carpetas, con el fin de utilizar aquellas más apropiadas para este proyecto. Como resultado a continuación se lista la estructura definida:

- **Carpeta 'data'**: Contiene las imágenes en crudo y procesadas que conforman el data set necesario para entrenar y validar el modelo
- **Carpeta 'docs'**: Contiene toda la documentación generada durante la realización del proyecto.
- **Carpeta 'models'**: Contiene los modelos entrenados y exportados para su uso en la arquitectura del sistema.
- **Carpeta 'notebooks'**: Contiene los diferentes notebooks de Python empleados para procesar las imágenes y construir/entrenar el modelo.
- **Carpeta 'reports'**: Contiene las diferentes predicciones obtenidas y exportadas por el modelo.
- **Carpeta 'src'**: Contiene los diferentes ficheros de código desarrollados para el funcionamiento del sistema.
- **Archivo '.gitignore':** Fichero utilizado por la herramienta GIT en el que se define la estructura de carpetas y ficheros que no van a ser tenidos en cuenta para la generación de nuevas versiones.
- **Archivo 'Readme.md':** Fichero que define la estructura de carpetas para facilitar la búsqueda de ficheros en el proyecto.
- **Archivo 'requirements.txt'**: Fichero que recoge las diferentes tecnologías y versiones empleadas para la realización del proyecto.

3.3.3 Tecnologías y dependencias

Para el desarrollo de este proyecto diferentes tecnologías han sido seleccionadas. En este apartado se expone y contextualiza cada una de ellas con el fin de establecer posteriormente su labor en el sistema resultado.

3.3.3.1 Git

Como se establece en puntos anteriores, Git es la tecnología seleccionada para establecer el control de versiones de este proyecto. Se trata de uno de los sistemas de gestión de versiones más utilizados de todo el mundo, dada su amplia gestión de los cambios y su rápido acceso a repositorios distribuidos como GitHub.

Al igual que otros gestores de versiones, Git proporciona a sus usuarios mejoras en el rendimiento, seguridad ante posibles pérdidas de información y flexibilidad para trabajar desde diferentes sistemas [10].

Debido a la posibilidad de acceso del historial completo desde *local*, la necesidad de red únicamente para la sincronización con repositorios remotos y la capacidad de registrar los cambios a nivel de cambio y no de fichero, Git ha sido la herramienta de gestión de versiones usada en este proyecto.

3.3.3.2 Conda - Miniconda

Uno de los factores más influyentes a la hora de desarrollar cualquier proyecto tecnológico es la gestión de entornos y dependencias. Muchos de los paquetes o librerías utilizados para la realización de proyectos poseen dependencias iguales o distintas que pueden generar incompatibilidades en el proceso de instalación. Además, encontrar, descargar e instalar las versiones específicas de cada una de estas dependencias y gestionarlas de forma adecuada puede suponer una gran carga de trabajo.

De esta forma surge Conda, un gestor de versiones que facilita la creación de diferentes entornos de trabajo y la instalación de las dependencias y librerías requeridas por sus usuarios, detectando en todo momento incompatibilidades y permitiendo conocer de forma sencilla todas y cada una de las versiones empleadas en el proyecto.

Para la implementación de Conda existen diferentes instaladores como Anaconda que traen consigo múltiples funcionalidades adicionales útiles para determinados usuarios. Sin embargo, en ocasiones estas funcionalidades extra suponen una carga de procesamiento innecesaria para el sistema contenedor que puede afectar en términos de recursos al desarrollo del proyecto [11].

Es por ello, que para el desarrollo de este proyecto se ha utilizado el instalador de Miniconda, que incluye las funcionalidades básicas para la creación de entornos y la gestión de las dependencias de Python que se verán en apartados posteriores.

3.3.3.3 Visual Studio Code

Visual Studio Code es un editor de código fuente gratuito desarrollado por la empresa Microsoft. Al tratarse de una herramienta gratuita dispone de acceso a una gran cantidad de *pluggins* y funcionalidades que los usuarios pueden instalar y utilizar según sus necesidades.

Esta herramienta trae integrado funcionalidades como el control integrado de Git, resaltado y recomendación de sintaxis e incluso compatibilidad con la selección de *kernels* y entornos de Conda. Además, existe una gran comunidad, formada por sus usuarios, que permite acceder a una gran cantidad de documentación y encontrar en la web la solución a muchas de las incidencias que pueden surgir durante el desarrollo del proyecto.

En este proyecto, esta herramienta será utilizada para el desarrollo de todos los códigos del sistema y para el acceso a los diferentes ficheros de texto.

3.3.3.4 Python

Python es un lenguaje de código abierto de propósito general ampliamente utilizado en el mundo del desarrollo software, ciencia de datos y *machine learning*.

Al ser uno de los lenguajes de programación más utilizados en la actualidad, dispone de una gran cantidad de librerías y dependencias creadas y documentadas por la comunidad, lo que facilita en gran medida el desarrollo de los proyectos y la resolución de incidencias que puedan surgir durante el mismo [12].

Sep 2022	Sep 2021	Change	Programming Language	Ratings	Change
1	2	^	Python	15.74%	+4.07%
2	1	v	C	13.96%	+2.13%
3	3		Java	11.72%	+0.60%
4	4		C++	9.76%	+2.63%
5	5		C#	4.88%	-0.89%
6	6		Visual Basic	4.39%	-0.22%
7	7		JavaScript	2.82%	+0.27%
8	8		Assembly language	2.49%	+0.07%
9	10	^	SQL	2.01%	+0.21%

Ilustración 4: Ranking TIOBE [13]

En este proyecto Python ha sido utilizado como lenguaje de desarrollo principal, siendo útil tanto para desarrollar los scripts de construcción y entrenamiento del modelo como para generar los scripts de generación y obtención de datos del sistema.

A continuación, se muestran recopiladas las librerías externas de Python empleadas para la realización de este proyecto:

3.3.3.4.1 PyAutoGUI

PyAutoGUI es un módulo de automatización para Python centrado en tareas simples como el movimiento del ratón, uso del teclado, cuadros de dialogo y capturas de pantalla en diferentes sistemas operativos.

En este proyecto, esta será la librería empleada para tomar las diferentes capturas de pantalla que serán enviadas y procesadas por el sistema para determinar si se está utilizando una red social en la imagen o si por el contrario no se ha detectado su uso.

3.3.3.4.2 OpenCV – cv2

OpenCV es una librería libre que establece la infraestructura necesaria para la realización de proyectos de visión artificial. Esta librería está compuesta de una gran cantidad de algoritmos que permiten desde identificar objetos hasta seguir el movimiento de los ojos.

En este proyecto se ha empleado el algoritmo de cv2 para la lectura y procesamiento de las imágenes necesarios para su ejecución e integración en el sistema con Keras y Apache Kafka.

3.3.3.4.3 Scikit-Learn

Scikit-learn es una librería de código abierto que recoge diferentes algoritmos de *machine learning*. Esta librería comprende desde algoritmos supervisados y no supervisados de aprendizaje automático hasta diferentes funciones de preprocesado de información, selección y evaluación de modelos [14].

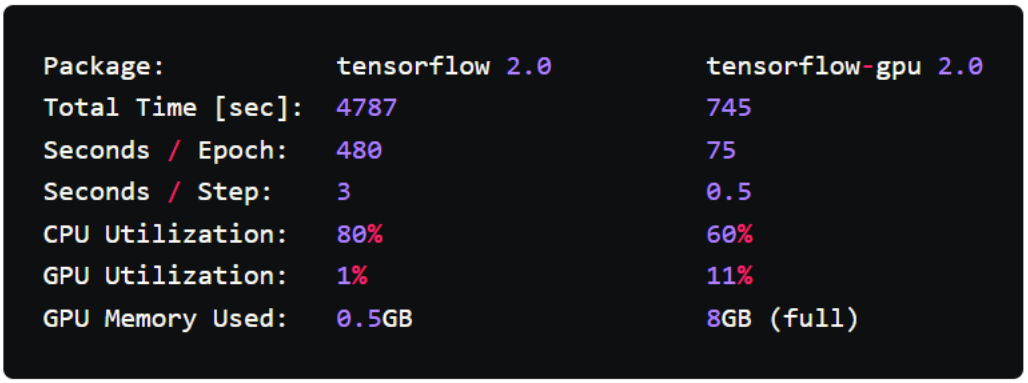
En el caso de este proyecto, Scikit-learn ha sido empleado para realizar la división del data set original en un data set de entrenamiento y de prueba de forma que la construcción y evaluación del modelo sea más eficiente.

3.3.3.4.4 Tensorflow y Keras

Keras es una librería de código abierto integrada dentro del motor de Google Tensorflow. Se compone de diferentes APIs, funciones y algoritmos centrados en facilitar la experimentación con redes de *Deep learning*.

Por otra parte, Tensorflow establece para Keras diferentes versiones que permiten su uso en sistemas paralelizables. De esta forma, para este proyecto se ha empleado la versión Keras-GPU, que permite paralelizar los cálculos durante el entrenamiento y la predicción mediante el uso de los cores de la tarjeta gráfica del equipo, logrando una reducción de tiempos bastante significativa.

A continuación, se puede ver un ejemplo de los resultados en tiempo obtenidos al comparar la ejecución de un algoritmo con Tensorflow de forma lineal y a través de la versión GPU.



Package:	tensorflow 2.0	tensorflow-gpu 2.0
Total Time [sec]:	4787	745
Seconds / Epoch:	480	75
Seconds / Step:	3	0.5
CPU Utilization:	80%	60%
GPU Utilization:	1%	11%
GPU Memory Used:	0.5GB	8GB (full)

Ilustración 5: Tensorflow CPU vs GPU

Como librería de construcción de redes, en este proyecto se ha empleado para la definición y construcción del modelo convolucional empleado para clasificar las imágenes en las diferentes clases establecidas.

3.3.3.4.5 Matplotlib

Esta librería de Python es una de las más utilizadas a la hora de representar diferentes elementos visuales como imágenes o gráficos.

En este proyecto, se ha empleado Matplotlib para visualizar las diferentes imágenes y predicciones obtenidas en el proceso de preprocesamiento, predicción y evaluación de los modelos. El elevado grado de customización de los elementos mostrados, ha hecho de esta librería un imprescindible para la comprensión de los resultados obtenidos.

3.3.3.5 Ubuntu (Máquina Virtual)

Dado que algunas de las tecnologías empleadas en este proyecto están diseñadas para funcionar únicamente en sistemas operativos Linux, en este proyecto se ha empleado una máquina virtual con el sistema Ubuntu 22.04.

En este sistema se han instalado y configurado las tecnologías de Java y Apache Kafka, y se ha establecido una red tipo puente de forma que el sistema virtual sea accesible desde el equipo anfitrión para el resto de componentes.

3.3.3.6 Apache Kafka

Apache Kafka es una plataforma *open source* distribuida para el almacenamiento y gestión de datos enviados en tiempo real. Este software permite generar y utilizar tópicos a los que los diferentes productores del sistema envían sus lecturas e información con el fin de que los sistemas de procesamiento o consumidores la procesen.

Este sistema permite establecer un paralelismo escalable de forma que el procesamiento de los datos siempre pueda ser llevado a cabo sin suponer retrasos.

En el caso de este proyecto, Apache Kafka ha sido instalado y configurado con el fin de gestionar las diferentes imágenes enviadas desde los productores y facilitar su procesamiento a los consumidores.

3.4 Generación y preprocesamiento de datos

En este apartado del documento se detallan los diferentes pasos y consideraciones seguidos para la construcción del sistema.

3.4.1 Construcción del data set

El sistema construido a lo largo de este proyecto pretende predecir si un usuario está utilizando aplicaciones de chat como WhatsApp a través de capturas de pantalla de su escritorio. Es por ello que el data set utilizado para entrenar el modelo constará de diferentes capturas de pantalla, almacenadas en formato PNG, que han sido tomadas manualmente en diferentes sistemas y escritorios.

Al utilizar el formato PNG, las imágenes no tienen perdida de información dado que el algoritmo de compresión empleado por este formato no la provoca. Como consecuencia, imágenes más pesadas han sido almacenadas [15].

Por otra parte, las imágenes utilizadas han sido interpretadas por el sistema por el espacio de color RGB. De esta forma, tres matrices (Una para el color rojo, otra para el color verde y otra para el color azul) serán las que conformen las imágenes [16].

Con el fin de aumentar el número de imágenes disponibles, se han realizado diferentes combinaciones usando variaciones de los siguientes aspectos y evitando en todo momento repetir imágenes ya incluidas con anterioridad:

- **Sistema operativo**: Posición de la barra de tareas, colores del sistema, posición y forma de las ventanas.
- **Fondos de escritorio**: Imágenes y colores distintos.
- **Resolución de la pantalla**: Las capturas de pantalla han sido tomadas usando las resoluciones 2560x1440 y 1920x1080 pixeles.
- **Aplicaciones abiertas**: Capturas de pantalla con múltiples aplicaciones abiertas tales como antivirus, navegadores web, aplicaciones de office, carpetas o editores de código, con variedad de colores y contenidos.
- **Tamaño de ventana**: Se han tomado capturas usando aplicaciones en modo pantalla completa y con diferentes variaciones de tamaño de ventana.
- **Colores y temas de aplicaciones**: Se han tomado capturas utilizando las mismas formas y consideraciones anteriores, pero modificando el tema (claro/oscuro) de las aplicaciones objetivo.

Como resultado, se ha obtenido un total de **304 capturas de pantalla** que han sido clasificadas en dos grupos:

- **Normal (114 imágenes)**: Conjunto de imágenes de escritorios que no se encuentran utilizando ninguna red social.
- **WhatsApp (190 imágenes)**: Conjunto de imágenes de escritorios que tienen abierta la aplicación de escritorio o la página web de la red social WhatsApp.

A continuación, se muestra un ejemplo de las capturas de pantalla seleccionadas y utilizadas durante el proyecto:

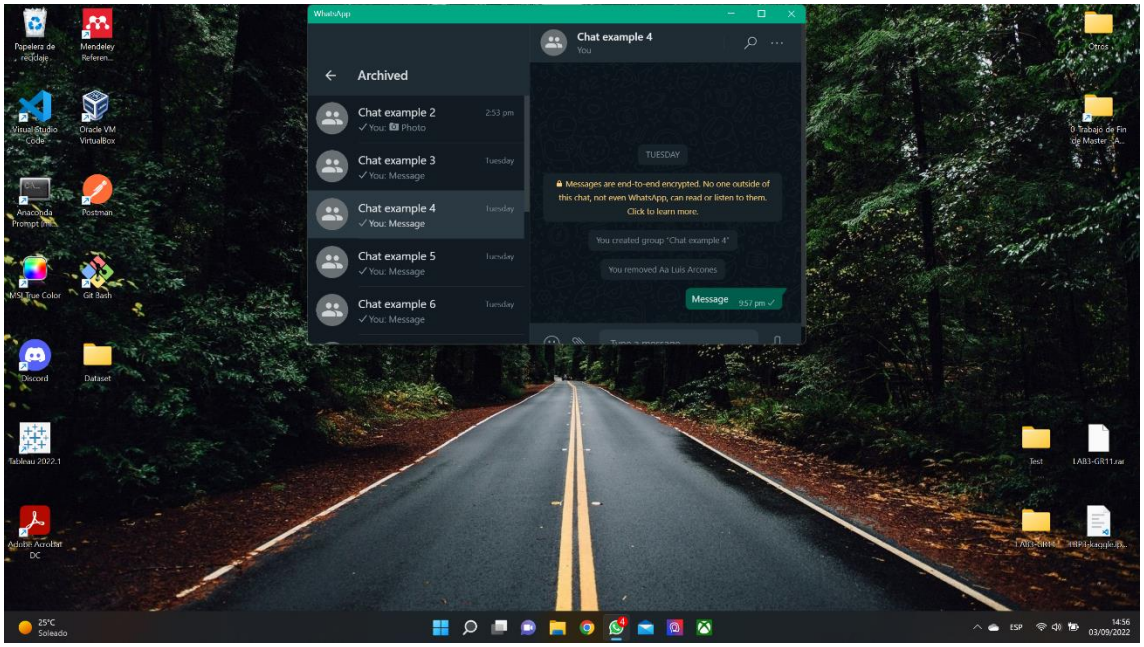


Ilustración 6: Ejemplo de imagen del data set

El data set construido en este apartado, al haber sido tomado de forma manual, se encuentra balanceado en número de imágenes disponibles para cada clase. Esto será algo tenido en cuenta a la hora de evaluar y optimizar el modelo de machine learning.

3.4.2 Preprocesamiento de las imágenes

Como se indica en el punto anterior, las diferentes imágenes que conforman el data set utilizado para este proyecto han sido tomadas utilizando diferentes resoluciones de pantalla. Estas resoluciones implican diferentes tamaños en la forma de las matrices y en el peso de los ficheros, que han sido determinantes a la hora de establecer ciertas características del sistema.

A continuación, se muestran las diferentes incidencias relacionadas con los datos detectadas durante el desarrollo del proyecto:

- **Recursos insuficientes durante el entrenamiento del modelo**: El gran tamaño de las matrices que componen las imágenes sumado al elevado número de operaciones realizadas durante el entrenamiento del modelo resultaron en múltiples errores relacionados con un espacio en memoria insuficiente.
- **Envió y recepción de imágenes**: El sistema conformado por el productor, consumidor e implementación de Apache Kafka no permitía enviar ni recibir las imágenes debido a su elevado peso, en ocasiones mayor al definido por la herramienta.

Con el objetivo de solucionar los problemas anteriores y conseguir unos resultados satisfactorios, en el proyecto se establece un procesamiento de las imágenes que reduce el tamaño de las capturas hasta alcanzar los 200 x 112 pixeles.

El tamaño seleccionado ha sido resultado de la realización de múltiples pruebas durante el entrenamiento del modelo y el envió de mensajes, utilizando en todo caso tamaños con una escala igual a la de las imágenes originales (16:9).

Para la reducción de las imágenes se ha desarrollado el notebook de Python *'1.0-PAP-Preprocesado_Imagenes.ipynb',* que mediante el uso de la librería cv2 aplica una interpolación cubica que mantiene la forma de las imágenes, evitando en la medida de lo posible grandes pérdidas de información.

Ilustración 7: Ejemplo imagen redimensionada

Como resultado, las diferentes imágenes procesadas han sido almacenadas en la carpeta '*data/processed*' con formato png, evitando la repetición del redimensionamiento en cada ejecución del modelo y reduciendo tiempos de ejecución.

3.4.3 Definición de los datos de entrenamiento y prueba

Con el objetivo de construir posteriormente un modelo de Deep learning y realizar en él un proceso de entrenamiento y validación de los resultados, se ha dividido el dataset en dos subconjuntos.

- **Conjunto de entrenamiento o train**: Este subconjunto está compuesto por el 80% del dataset construido anteriormente, y será el utilizado a la hora de entrenar el modelo para que este aprenda y modifique sus pesos.
- **Conjunto de validación/pruebas o test**: Este subconjunto se encuentra compuesto por el 20% restante del dataset, y es el utilizado para evaluar el proceso de entrenamiento del modelo (Al ser imágenes no utilizadas durante el entrenamiento).

Para evitar sesgos en el proceso de evaluación del modelo, se ha comprobado que no existan imágenes duplicadas entre ambos datasets.

3.5 Modelado y optimización

3.5.1 Selección y definición del modelo

El modelo de este proyecto es el encargado de clasificar las imágenes en función de si detecta el uso herramientas de chat o no. Dado que el data set construido únicamente posee dos tipos de imágenes, el modelo se entrenará con el objetivo de predecir dos clases distintas: No-chating y WhatsApp.

El tipo de red seleccionado para la realización de este proyecto es la convolucional o CNN. Este tipo de redes permiten extraer en cada capa diferentes características y formas sobre las imágenes (datos en forma de tabla) que reciben, lo cual las convierte en redes especialmente eficaces a la hora de detectar patrones y predecir clases sobre este tipo de datos.

La implementación del modelo ha sido realizada a través de notebooks de Jupyter. Estos notebooks han sido desarrollados utilizando el software de Visual studio code y han facilitado la ejecución y análisis de los resultados obtenidos.

Al tratarse de un software que permite ejecutar código de Python por celdas (guardando las variables de cada celda de forma temporal) la re ejecución de transformaciones, entrenamientos y generación de gráficas no ha requerido de re ejecuciones completas del código, reduciendo enormemente el tiempo empleado para el análisis y la obtención de resultados.

Keras ha sido la librería principal utilizada para el desarrollo del modelo, gracias a su gran documentación y cantidad de opciones que ofrece para el desarrollo de redes convolucionales. De esta forma, de aquí en adelante, todos y cada uno de los elementos explicados se encontrarán relacionados con las funciones de esta librería.

Los modelos convolucionales utilizan diferentes tipos de capas y meta-parámetros. A continuación, se encuentran definidos aquellos que han sido utilizados en este proyecto:

- **Capa de convolución 'Conv2D'** – Capa que crea un kernel de convolución y lo aplica sobre las imágenes de entrada, dando como resultado un conjunto de imágenes de salida para cada imagen [17].
    - **Meta-parámetro 'filters'** – Número de filtros de salida de la convolución.
    - **Meta-parámetro 'kernel_size'** – Indica la altura y anchura de la ventana de convolución aplicada a las imágenes.
    - **Meta-parámetro 'padding'** – Indica si la salida de la convolución debe ser ajustada para igualar el tamaño de la entrada.
    - **Meta-parámetro 'activation'** – Función de activación utilizada.
- **Capa de pooling 'MaxPooling2D'** – Reduce las dimensiones espaciales de la muestra de entrada.
    - **Meta-parámetro 'pool-size'** – Tamaño de la ventana.

  - **Meta-parámetro 'data_format'** – Establece el orden de las dimensiones en la entrada.
- **Capa 'Flatten'** – Reduce la dimensionalidad de la entrada para conectar las capas de convolución con las del perceptrón multicapa.
- **Capa 'Dense'** – Crea una capa de neuronas similar a la de un perceptrón multicapa.
  - **Meta-parámetro 'activation'** – Función de activación utilizada.

A continuación, se puede ver el modelo construido mediante el uso de Visual studio, Python y la librería de Keras.

```
learning_rate = 0.0001
batch_size = 16
```

```
#Creacion de la red
model = Sequential()

#Parte de la red CNN

model.add(Conv2D(120, kernel_size=(3, 3), padding="same", input_shape=(112, 200, 3), activation="relu"))
model.add(Conv2D(120, kernel_size=(3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_first")) #Capa de pooling

model.add(Conv2D(120, kernel_size=(3, 3), padding="same", activation="relu"))
model.add(Conv2D(120, kernel_size=(3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_first")) #Capa de pooling


#Parte de la red MLP

model.add(Flatten()) #Unimos la CNN con el MLP

model.add(Dense(500, activation="relu"))
model.add(Dense(1000, activation="relu"))


model.add(Dense(len(tipos), activation = "softmax"))

model.compile(loss="categorical_crossentropy", optimizer=optimizers.Adam(learning_rate=learning_rate), metrics=["accuracy"])
#model.save('tfm_model.h5')
print("Terminado")
```

Ilustración 8: Modelo construido en Python

Con respecto al entrenamiento del modelo, existen diferentes parámetros que son importantes y que establecen la forma en la que el modelo aprenderá a partir de los datos de entrenamiento. Estos parámetros son:

- **Epochs** – Las epochs son el número de iteraciones que va a realizar el modelo sobre el conjunto de datos de entrenamiento en el proceso de entrenamiento del modelo. Si este número es muy elevado, se puede dar el caso de un sobre entrenamiento del modelo. Sin embargo, si el valor es muy bajo, el modelo obtenido no ofrecerá buenos resultados.
- **Batch size** – Representa el número de ejemplos que serán utilizados en cada iteración para entrenar el modelo.
- **Optimizador** 'Adam'- El optimizador Adam es un método que emplea el descenso de gradiente estocástico y que se basa en la estimación adaptativa de momentos de primer y segundo orden.

Dentro de este optimizador se puede definir el meta-parámetro '**learning_rate'** que representa el aprendizaje realizado sobre los pesos en cada una de las iteraciones.

Las métricas empleadas para la evaluación de este modelo serán el loss y el accuracy. Estas métricas permiten saber el estado de un modelo en base al número de aciertos/fallos realizados por el mismo durante el entrenamiento. Estas métricas han sido seleccionadas debido a que el dataset empleado para el entrenamiento del modelo se encuentra balanceado entre clases, es decir, que tiene aproximadamente el mismo número de imágenes de cada clase. A continuación, se exponen las métricas utilizadas en el modelo:

- **Loss** – Definida como 'CategoricalCrossEntropy' calcula la entropía cruzada entre la categoría de los datos de entrada del conjunto de entrenamiento y el de las predicciones realizadas por el modelo. Cuanto más cercano sea este valor a 0, mejor será el modelo.
- **Accuracy** – Calcula en forma de porcentaje el número de predicciones hechas que coinciden con la categoría establecida en el data set de entrenamiento con respecto al total de imágenes entrenadas. Cuanto más cercano a 1, más se habrá adaptado el modelo al data set y mejores predicciones hará.
- **Val_loss** - Definida como 'CategoricalCrossEntropy' calcula la entropía cruzada entre la categoría de los datos de entrada del conjunto de prueba y el de las predicciones realizadas por el modelo. Cuanto más cercano sea este valor a 0, mejor será el modelo.
- **Val_Accuracy** - Calcula en forma de porcentaje el número de predicciones hechas que coinciden con la categoría establecida en el data set de prueba con respecto al total de imágenes entrenadas. Cuanto más cercano a 1, más habrá aprendido el modelo y mejores predicciones hará con imágenes no incluidas en el proceso de entrenamiento.

Una vez establecido el data set y seleccionado el tipo de modelo, los datos numéricos que componen las imágenes han sido normalizados dividiéndose entre 255, el cual es el valor máximo en las matrices RGB, de forma que todos los valores queden comprendidos entre 0 y 1.

Además, se ha establecido un método de early stopping que detenga el entrenamiento cuando los valores de validación empeoren sus resultados entre iteraciones, con el fin de evitar el sobre entrenamiento del modelo y obtener el mejor resultado posible para la configuración probada.

3.5.2 Definición de casos de prueba y optimización del modelo

El proceso de optimización de cualquier modelo que emplee algoritmos de inteligencia artificial requiere de la definición de diferentes casos de pruebas sobre los que se almacena y evalúa las métricas previamente indicadas.

En este apartado se encuentran definidos los diferentes casos que se han probado junto con el proceso seguido para la definición de los mismos, tras lo cual, en el apartado de resultados, se podrán ver los valores obtenidos.

Es importante tener en cuenta que, con el fin de obtener los resultados más fiables posibles, se ha establecido una validación cruzada de tres ejecuciones, calculando la media de cada una de las métricas establecidas y registrándolas en la tabla mostrada en el apartado de resultados. De esta forma se reduce el efecto causado por la aleatoriedad de los pesos iniciales del modelo que se genera con cada ejecución y mostrando así unos resultados promedio que permitan replicar el modelo en el futuro con resultados similares.

Por otra parte, al haber definido un proceso de early stopping, el número de epochs mostrados en la tabla de resultados también será resultado de la media entre las tres ejecuciones de la validación cruzada.

3.5.2.1 Optimización de la arquitectura del modelo

El primer componente que ha sido seleccionado para la optimización del modelo es la arquitectura. En este apartado se indicarán los casos de prueba seleccionados y llevados a cabo para la optimización del modelo, así como las consideraciones necesarias.

Dado que la optimización del modelo se puede realizar a través de la modificación de los valores de los múltiples parámetros anteriormente explicados, en este apartado se han establecido las pruebas relativas a las capas y neuronas que componen el modelo.

Para la optimización de dicha arquitectura se han fijado los siguientes parámetros con el fin de no variar el funcionamiento del modelo y ser capaces de analizar únicamente la influencia de las diferentes configuraciones de capas y neuronas.

- **Capas de convolución**
  - Tamaño del kernel
  - Padding
  - Función de activación de las neuronas
- **Capas de Pooling**
  - Pooling
- **Capas del MLP**
  - Función de activación
- **Learning rate**

A continuación, se definen los casos de prueba que se han llevado a cabo en el modelo para su optimización:

- Se han probado modelos con un número de capas de pooling entre 1 y 4.
- El número de capas de convolución por cada capa de pooling ha sido probado utilizando entre una y dos capas de convolución.
- El número de capas utilizados en la parte del modelo correspondiente al MLP ha sido fijado a 2 capas ocultas.
- Se ha probado con un número bajo de filtros que a medida que se iban seleccionando las configuraciones con mejores resultados se han aumentado.
- Se han probado configuraciones con un mayor número de filtros en las primeras capas de convolución y del MLP que en las segundas y viceversa.

De esta forma, las arquitecturas que mejores resultados ofrecían ha sido aquellas seleccionadas sobre las que aplicar los cambios de los puntos siguientes.

3.5.2.2 Optimización de los meta-parámetros del modelo

Una vez establecidas las pruebas realizadas para la arquitectura y obtenidos los resultados, se ha seleccionado aquel con mejores valores de accuracy y de loss.

Sobre el modelo seleccionado, se han fijado y/o seleccionado los valores predeterminados del resto de meta-parámetros y se han establecido las pruebas necesarias para optimizar el meta-parámetro 'learning-rate'.

Para ello se ha establecido el modelo con la arquitectura seleccionada y se han realizado ajustes sobre le meta-parámetro. A continuación, se muestran los valores tomados por el *learning rate* que serán probados:

- 0.001
- 0.0005
- 0.0001
- 0.00005
- 0.00001

Una vez obtenidos los resultados, se ha seleccionado el valor que ofrecía los mejores resultados de los y accuracy y se ha establecido junto con la arquitectura previamente seleccionada como el modelo resultado del proyecto.

Finalmente, y tras haber seleccionado la configuración de la arquitectura y de los meta-parámetros que mejores resultados ha ofrecido, se ha deshabilitado el control de

entrenamiento del early stopping y establecido un número de 50 epochs, con el fin de representar y analizar la evolución de las métricas durante la fase de entrenamiento.

Los resultados obtenidos de la realización de las pruebas definidas anteriormente se encuentran en el capítulo 4 de este mismo documento, en el cual también se analizan los valores obtenidos y se expone el modelo seleccionado.

3.5.3 Persistencia del modelo

Una construido, entrenado y optimizado el modelo convolucional, este ha sido persistido de forma que posteriormente pudiese ser utilizado por el sistema de Apache Kafka o por cualquier usuario que desee replicar los resultados obtenidos en este proyecto.

Para ello, dos ficheros han sido creados y almacenados en la carpeta '*models*' establecida previamente en la estructura de carpetas del proyecto:

- **Model_structure.json** – Contiene la estructura de capas y meta-parámetros del modelo construido.
- **Model_weights.h5** – Contiene los pesos ya entrenados resultantes del proceso de entrenamiento y optimización del modelo.

De esta forma, cualquier sistema que desee emplear este modelo o usuario que quiera probar los resultados únicamente necesitará generar un script de Python en el que importará la librería de JSON para leer el fichero y la función de Keras 'model_from_json' para convertir el contenido en un modelo funcional de la siguiente forma:

```
#Load trained model
json_file = open('C:/Users/pablo/Documents/Master Big Data/0-Trabajo de Fin de Master/Repositorio Git/TFM-Big-Data/models/model_structure.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("C:/Users/pablo/Documents/Master Big Data/0-Trabajo de Fin de Master/Repositorio Git/TFM-Big-Data/models/model_weights.h5")
```

Este proceso ha sido seguido en la construcción del consumidor de Apache Kafka que se verá en los siguientes apartados[18].

3.6 Construcción del sistema con Apache Kafka

Apache Kafka es el software encargado de la transmisión y almacenamiento de las imágenes e información en el sistema. Este software dispone de la posibilidad de ejecutarse de forma distribuida en varios equipos para mejorar sus capacidades; sin embargo, para la realización de este proyecto, y debido a los recursos disponibles, únicamente se ha instalado en un solo sistema.

Dadas las necesidades de este software, una máquina virtual con el sistema operativo Ubuntu ha sido instalada y configurada. La máquina virtual necesita poder ser accedida

desde el sistema host. Para ello, ha sido necesario configurar sus conexiones a modo puente, de forma que las direcciones IP de ambos sistemas sean diferenciadas, y asegurando en todo momento que los sistemas pueden visualizarse entre sí a través del comando ping.

Una vez establecida la máquina virtual, se ha descargado e instalado Java (dependencia) y el software de Apache Kafka.

A la hora de ejecutar y arrancar el servicio de Kafka, es necesario arrancar previamente el servicio de Zookeper, ya que es el encargado de coordinar y gestionar las diferentes instancias de Kafka, a pesar de que en este proyecto únicamente se emplea una.

Para iniciar el sistema, se ejecutan los siguientes comandos en orden y en ventanas diferentes del terminal:

1. *bin/zookeeper-server-start.sh config/zookeeper.properties*
2. *bin/kafka-server-start.sh config/server.properties*

Una vez levantados los sistemas un tópico de nombre 'test_topic' fue creado, siendo este el empleado por el bróker como punto de almacenamiento y como punto de escritura y lectura de la información para el resto de componentes del sistema.

Una vez creado y configurado el bróker de Kafka, un productor y un consumidor Python fueron construidos para la escritura en tiempo real de las imágenes tomadas y el uso del modelo construido para la predicción de imágenes, como se puede ver en el apartado de arquitectura de este mismo documento.

3.6.1 Productor

El productor de este sistema es el encargado de tomar las capturas de pantalla, ajustarlas a las necesidades del sistema establecidas en el preprocesamiento y enviarlas al tópico del bróker correspondiente con el fin de garantizar su procesamiento.

Es por ello que es necesaria su instalación y correcto funcionamiento en cada uno de los sistemas que se desee que envíen imágenes para su clasificación, teniendo tantos productores como equipos que lo utilicen, y necesitando en todo momento que los equipos que utilicen el sistema se encuentren en la misma red.

Para lanzar el proceso es necesario usar la consola, indicando el nombre del tópico al que se desea enviar y el número de segundos que pasará entre las capturas de pantalla. Una vez lanzado el proceso, un bucle infinito tomará las capturas, las redimensiona y las envía en formato JSON al bróker de Kafka, dejando entre captura y captura el tiempo indicado.

3.6.2 Consumidor

El consumidor de este proyecto será el encargado de importar el modelo previamente exportado, tomar las diferentes capturas de pantalla almacenadas en Kafka y predecir su clase, almacenándolas y dejando las evidencias correspondientes en el sistema.

A diferencia del productor, el número de consumidores que se utilizarán en el sistema será igual al número de particiones establecidas para el tópico de Kafka. De esta forma, se garantiza el paralelismo en el procesamiento de la información y la escalabilidad del sistema.

Por otra parte, y al igual que ocurría con los productores, estos deben ser lanzados por consola de comandos, con la excepción de que estos únicamente necesitan recibir como parámetro el nombre del tópico.

Las diferentes predicciones realizadas por los consumidores a través del modelo son almacenadas en las carpetas contenidas dentro del directorio '*reports/predictions*'. En esta carpeta se almacenan otras carpetas que corresponden a las clases definidas en el modelo, almacenando cada una de las imágenes en su clasificación correspondiente. Además, el consumidor anotará en el fichero Log.log, contenido en la misma carpeta, los porcentajes correspondientes a la predicción realizada. [19, 20]

Capítulo 4. Pruebas y resultados

En este apartado se muestran los resultados obtenidos de la realización de este proyecto. Estos resultados se encuentran divididos en dos partes, que corresponden al modelo entrenado y al sistema implementado con Apache Kafka.

4.1 Resultados obtenidos en la optimización del modelo

Los casos de prueba establecidos en el capítulo 3 de este documento, recogen cada una de las pruebas que se han llevado a cabo para la optimización del modelo.

4.1.1 Arquitectura del modelo

Uno de los elementos a configurar más importantes y que ha determinado los resultados obtenidos por el modelo es la arquitectura. Los diferentes tipos de capas, números de neuronas y combinaciones de capas hacen que su configuración pueda tomar infinitos valores.

Con el fin de construir una arquitectura para el modelo que sea adecuada para el problema, se han realizado múltiples ejecuciones en las que se ha utilizado el loss, accuracy, validation los y validation accuracy para medir la efectividad del modelo. Además, con el fin de comparar únicamente los efectos de las modificaciones de la arquitectura del modelo y no de los meta-parámetros, se han fijado los valores de algunos de estos meta-parámetros a los siguientes:

- **Capas de convolución**
  - Tamaño del kernel: 3x3
  - Padding: ‘Same’
  - Función de activación de las neuronas: ‘Relu’
- **Capas de Pooling**
  - Pooling: 2x2
- **Capas del MLP**
  - Función de activación: ‘Relu’ (Última capa con valor ‘softmax’)
- **Learning rate:** 0.0001

Una vez fijados estos valores, se han realizado múltiples ejecuciones en las que se han anotado las métricas indicadas anteriormente con el fin de comparar las arquitecturas. A continuación, se muestra una tabla con los diferentes valores obtenidos de la validación cruzada de cada una de las combinaciones:

Epochs	Numero de capas de convolución por grupo	Neuronas por capa de convolución	Neuronas por capa de convolución 2	Neuronas por capa de convolución 3	Neuronas por capa de convolución 4	Neuronas MLP 1	Neuronas MLP 2	Loss	Accuracy	Validation Loss	Validation Accuracy
3	1	5				120	120	19,6530	0,7393	32,3621	0,5410
4	1	5	5			120	120	3,1438	0,8546	9,6326	0,6120
4	1	5	5	5		120	120	5,5215	0,7627	5,4973	0,6393
7	1	5	5	5	5	120	120	1,3215	0,8039	2,6566	0,6940
3	2	5				120	120	5,1592	0,7174	18,5750	0,5574
4	2	5	5			120	120	3,1146	0,7284	24,9972	0,7268
2	2	5	5	5		120	120	0,5307	0,7709	0,9210	0,6284
7	2	5	5	5	5	120	120	0,1807	0,9520	0,5082	0,6014
4	2	5	5			500	1000	1,8926	0,7709	3,4581	0,7213
4	2	5	5			1000	500	0,6626	0,8820	1,5651	0,7049
5	2	30				120	120	4,3189	0,8505	15,8884	0,7158
4	2	30	30			120	120	4,5759	0,7463	1,6706	0,7432
3	2	30	30	30		120	120	0,6160	0,7407	0,7840	0,6175
6	2	30	30	30	30	120	120	0,3977	0,8340	0,6109	0,6885
5	2	30	30			500	1000	0,2861	0,9204	1,1173	0,7869
3	2	30	30			1000	500	12,7305	0,7517	11,8192	0,6503
4	2	10	20			500	1000	6,3193	0,7421	13,6144	0,5410
4	2	20	10			500	1000	0,5284	0,8732	2,7025	0,6284
4	2	30	60			500	1000	6,0077	0,8258	8,5185	0,7541
6	2	5	15	30	60	120	120	0,2958	0,8779	0,5373	0,6940
6	2	60	60			500	1000	0,0452	0,9890	1,1136	0,8197
5	2	120	120			500	1000	0,0603	0,9822	0,3500	0,8525
6	2	60	120			500	1000	0,0723	0,9712	0,6168	0,8361
4	2	120	60			500	1000	0,4899	0,8971	1,5246	0,7158

Ilustración 9: Pruebas modelo: Arquitectura

Como se indica en los casos de prueba establecidos, las primeras pruebas realizadas sobre el modelo se han hecho empleando una capa de convolución por cada capa de pooling, pocos filtros en cada capa de convolución y un valor fijo en las capas de MLP.

Con estos valores, se han ejecutado pruebas con hasta 4 capas ocultas, en las que se puede ver que los mejores resultados ofrecidos son los proporcionados con 3 y 4 capas.

Sin embargo, a la hora de aumentar el número de capas de convolución por capa de pooling, se puede apreciar como los resultados mejoran significativamente a 0.7268 en cuanto al accuracy de validación, además de cambiar la configuración optima a dos capas de pooling en vez de las 3 y 4 anteriores. De esta forma y en adelante, se han probado diferentes configuraciones aplicadas sobre redes con las mismas características (Número de capas de convolución por capa de pooling y número de capas de pooling).

Como resultado de las pruebas anteriores, se ha aumentado el número de neuronas en el modelo MLP y en el modelo convolucional, intercalando entre ejecuciones el mayor y menor número de neuronas en la primera y segunda capa, con el fin de observar los resultados obtenidos.

Una vez realizadas las siguientes pruebas, se ha podido ver como utilizando en la capa del perceptrón un menor número de neuronas con respecto a la segunda capa, se obtenían mejores resultados, por lo que esta ha sido la configuración del perceptrón seleccionada.

Sin embargo, las diferentes pruebas realizadas sobre las capas de convolución han llevado a tomar los mismos números de filtros en la primera y segunda capa de pooling.

4.1.2 Búsqueda de meta-parámetros

Una vez analizados los resultados obtenidos de las diferentes pruebas realizadas con la arquitectura, se ha procedido a hacer una búsqueda de meta-parámetros.

El meta-parámetro que se ha analizado y con el que se han realizado diferentes pruebas es el learning-rate que se había fijado en el punto anterior a un valor de 0.0001. A continuación, se muestran los diferentes valores que se han probado, con los resultados correspondientes:

Epochs	Learning rate	Loss	Accuracy	Validation Loss	Validation Accuracy
2	0,001	0,5283	0,7407	0,6443	0,7924
3	0,0005	0,1667	0,9369	0,5115	0,8252
5	0,0001	0,0603	0,9822	0,3500	0,8525
3	0,00005	0,1311	0,9602	0,6296	0,8033
2	0,00001	2,0275	0,7503	15551,0110	0,3770

Ilustración 10: Pruebas Modelo: Meta-parámetros

Como se puede ver en los resultados de la imagen anterior, se han probado los valores previamente definidos en los casos de prueba. Sin embargo, no parece que ninguno de los valores haya mejorado con respecto al valor inicial tomado y fijado en el modelo, por lo que este ha sido el valor seleccionado como optimo.

Se puede ver marcado en amarillo las configuraciones que han sido tomadas como optimas, ya que han sido aquellas que han ofrecido los mejores resultados.

Una vez seleccionadas las configuraciones optimas, se ha exportado su estructura en formato json y sus pesos en formato h5, de forma que pueda ser importada y utilizada por el sistema implementado en Apache Kafka.

4.1.3 Análisis del entrenamiento del modelo con la configuración óptima

Como se establece en el apartado de casos de prueba del capítulo 3, y con el objetivo de analizar el comportamiento del modelo durante la fase de entrenamiento, se ha deshabilitado la funcionalidad de early stopping y se ha procedido a re entrenar el modelo.

Una vez hecho esto, se han recogido los históricos de los y accuracy resultado de los diferentes epoch de entrenamiento, obteniendo los siguientes resultados:

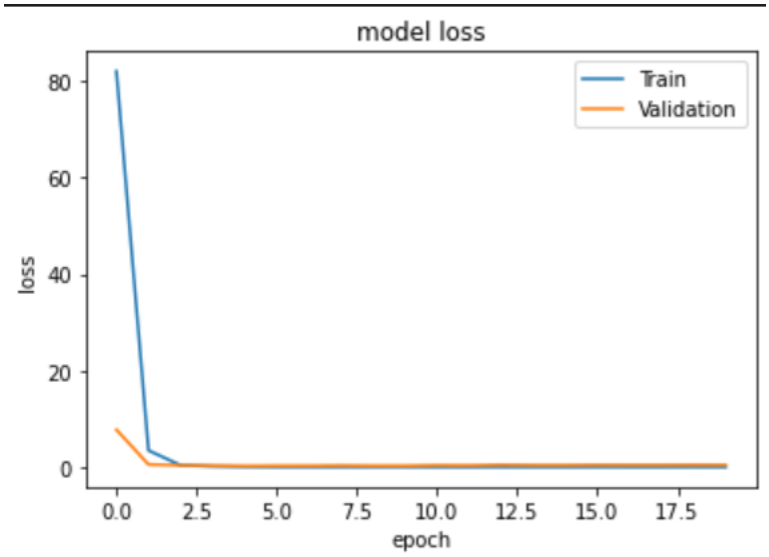


Ilustración 11: Resultado del modelo - Loss function

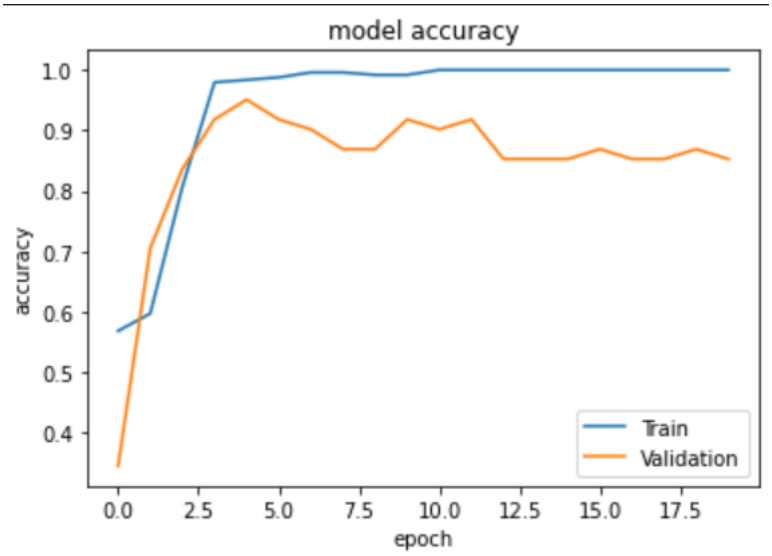


Ilustración 12: Resultado del modelo - Accuracy function

Como se puede ver en las gráficas anteriores, el modelo muestra unos mejores resultados de los que ofrecía en las tablas de los dos puntos anteriores. En estos gráficos se observa como el modelo toma valores similares a los mostrados anteriormente para el loss y el accuracy de los datos de entrenamiento, pero también toma valores de hasta 0.95 para el accuracy de validación entre la tercera y la cuarta epoch.

A partir de dicho punto, el modelo comienza a empeorar en cuestión del accuracy de validación, al mismo tiempo que representa una mejora asintótica con respecto a los datos de entrenamiento, considerándose esto en un sobre entrenamiento.

Debido a este análisis, y gracias a los mostrados obtenidos en el análisis anterior, el modelo ha sido establecido con un número de 4 epochs, dado que el early stopping estaba proporcionando una epoch de más que suponía una perdida de casi un 0.07 en el accuracy del modelo.

4.2 Resultados del modelo óptimo

Tras las múltiples ejecuciones y análisis mostrados en el punto anterior, se ha seleccionado aquel modelo que ofrecía los mejores resultados en cuando a las métricas de accuracy y validation_accuracy (ya marcadas en amarillo en las tablas).

A continuación, se muestra el diagrama de capas que componen el modelo seleccionado, mostrando desde las capas de la red convolucional hasta las capas del perceptrón multicapa.

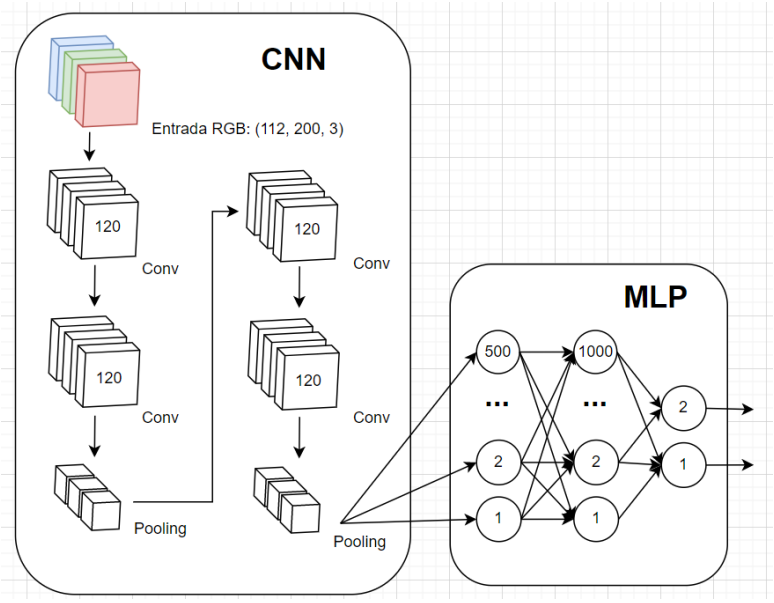


Ilustración 13: Diagrama modelo neuronal

El modelo mostrado proporcionaba un accuracy de 0.9822 al utilizar las imágenes del data set de entrenamiento y un accuracy de 0.8525 con las imágenes del data set de test.

Con el fin de analizar los resultados obtenidos, se han tomado y visualizado las predicciones realizadas sobre algunos de las imágenes del data set de test, mostrando los siguientes resultados:

1. **Imagen de prueba 1**

Esta captura de pantalla muestra un un escritorio que se encuentra utilizando Google drive, sin tener la aplicación de WhatsApp abierta y la web de WhatsAppWeb en uso.

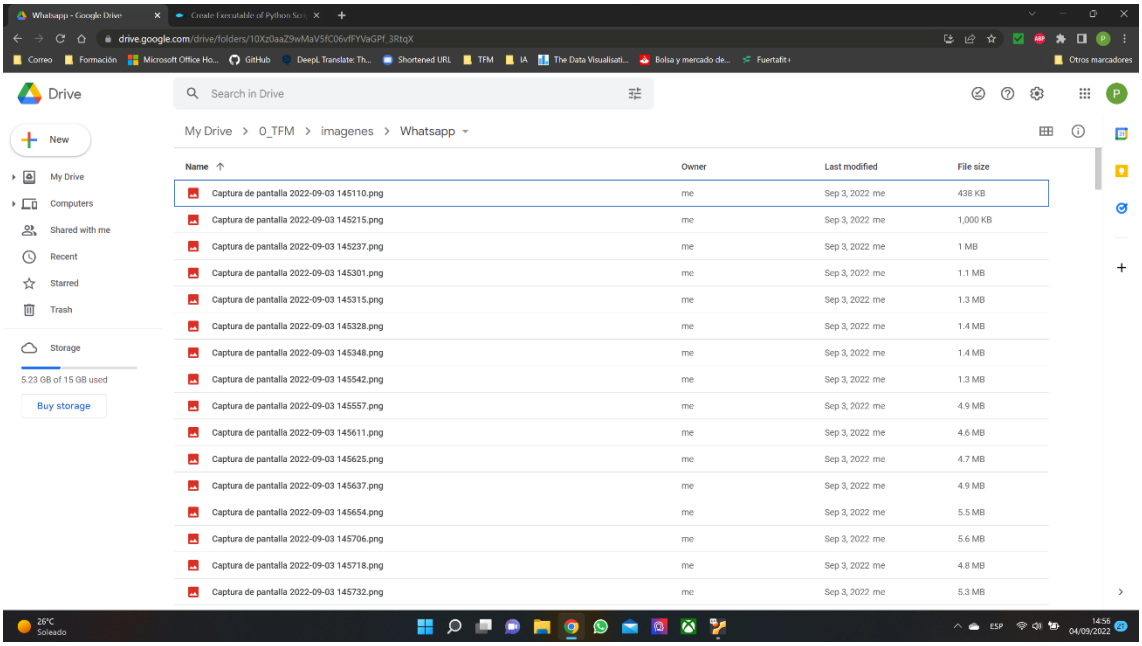


Ilustración 14: Predicción del modelo - imagen 1

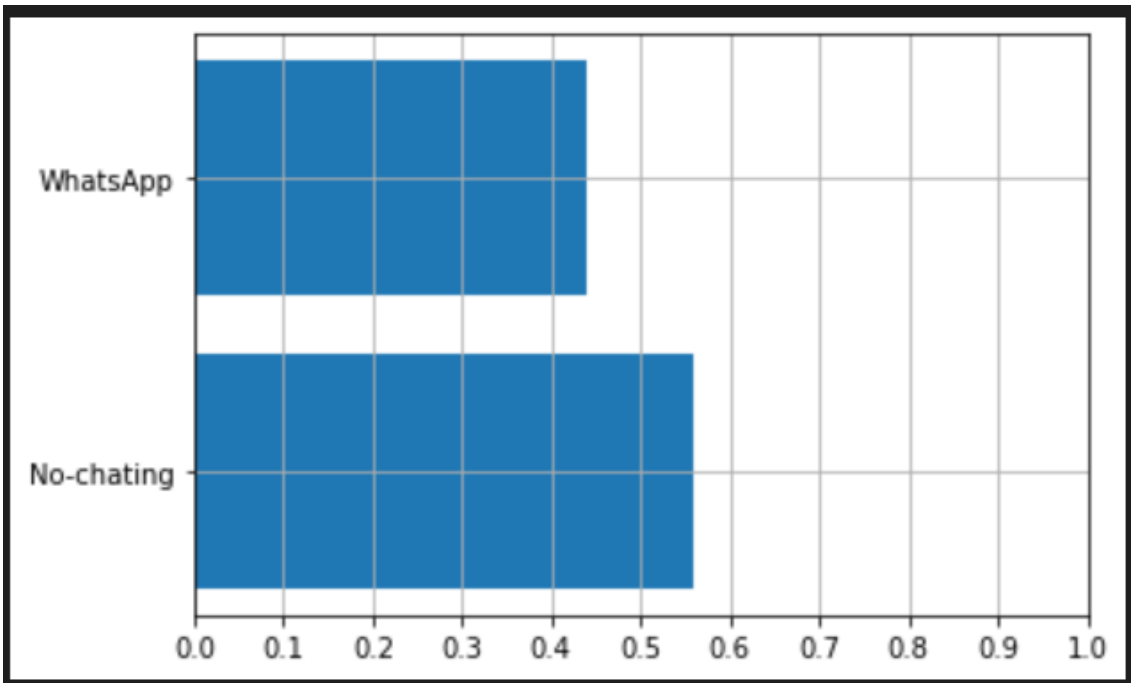


Ilustración 15: Predicción del modelo - resultado 1

Como se puede ver en el resultado obtenido por el modelo de la imagen anterior, el modelo clasifica esta imagen como 'No-chating', identificando así que ninguna herramienta de chat está actualmente en uso. Sin embargo, puede apreciarse que el valor que representa la clase de 'WhatsApp' se aproxima bastante al valor asociado a la clase de 'No-chatting', por lo que el modelo no está haciendo una diferenciación clara.

2. Imagen de prueba 2

Esta imagen muestra un escritorio que se encuentra navegando por la web, sin utilizar WhatsApp Web ni la aplicación de escritorio de WhatsApp.

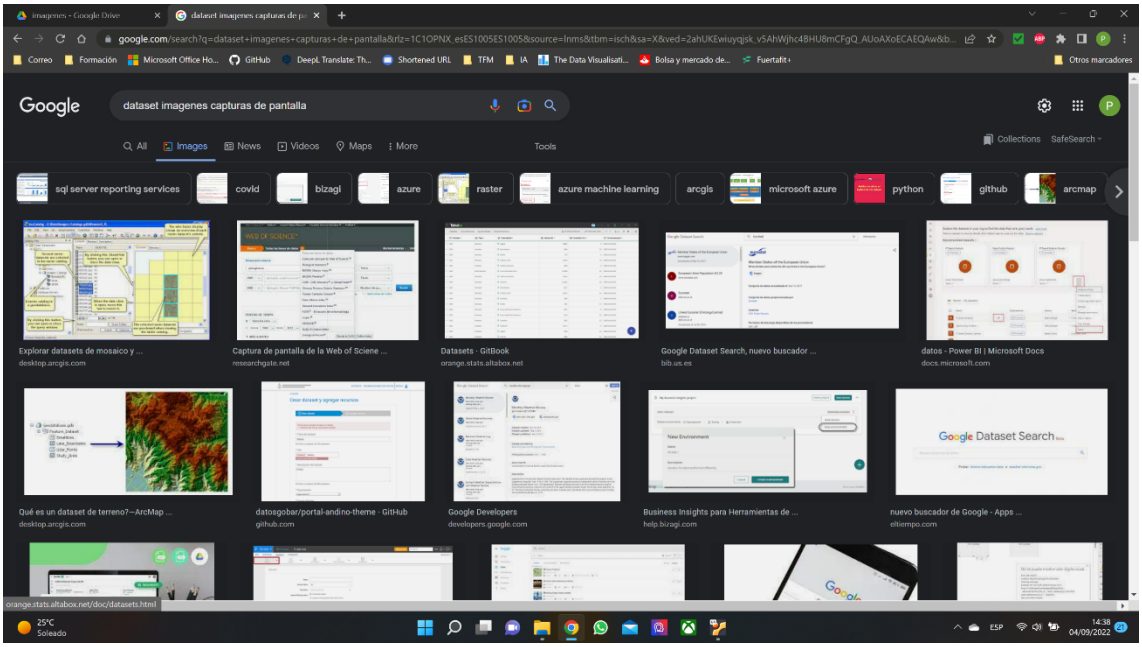


Ilustración 16: Predicción del modelo - imagen 2

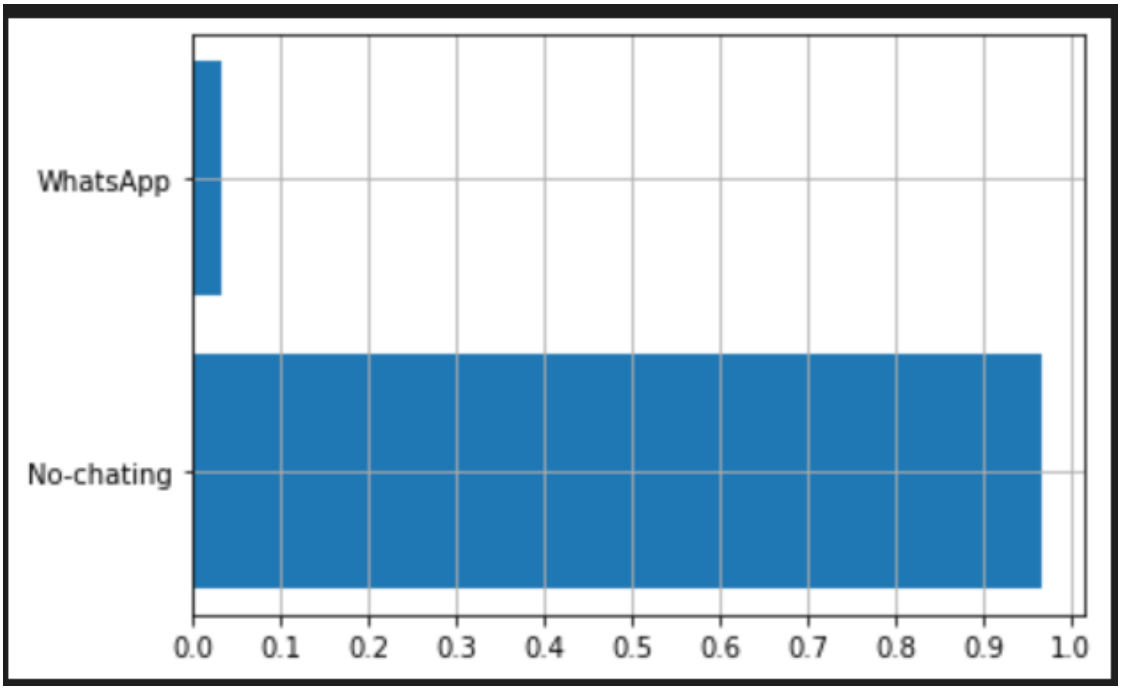


Ilustración 17: Predicción del modelo – predicción 2

Como se puede ver en la imagen anterior, los resultados mostrados por el modelo es una clara clasificación de la imagen como 'No-chatting', identificando claramente que no se encuentra utilizando ninguna herramienta de chat.

3. Imagen de prueba 3

En esta captura de pantalla se puede apreciar una ventana con la aplicación de Whatsapp Web con tema oscuro abierta sobre otra ventana del navegador de fondo blanco, en un escritorio Linux.

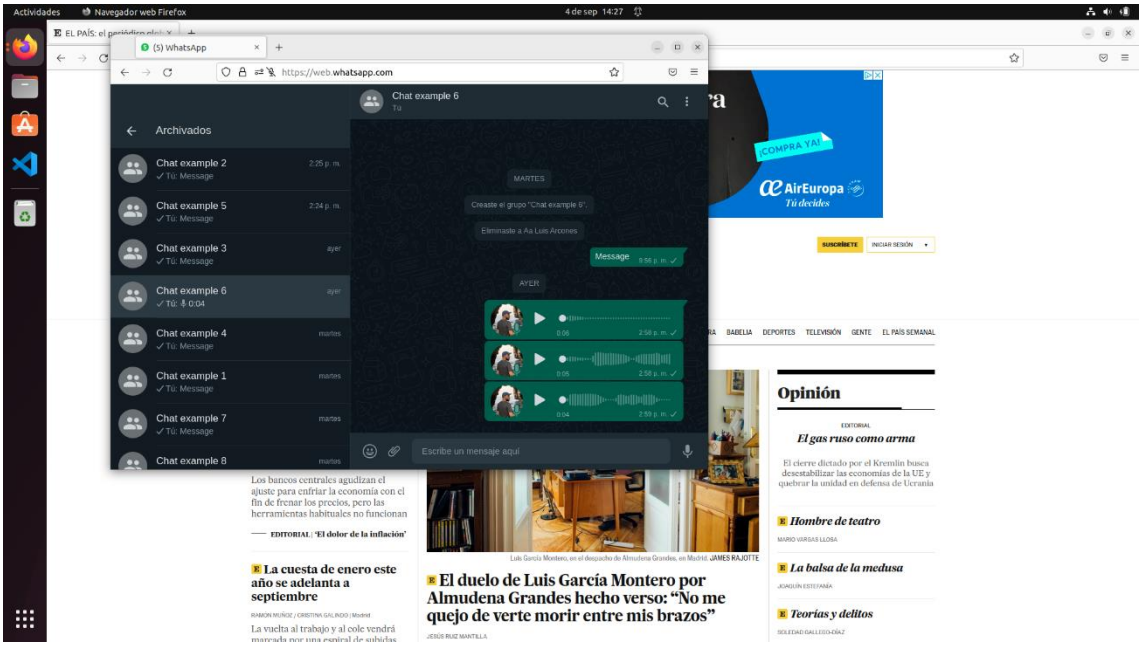


Ilustración 18: Predicción del modelo - imagen 3

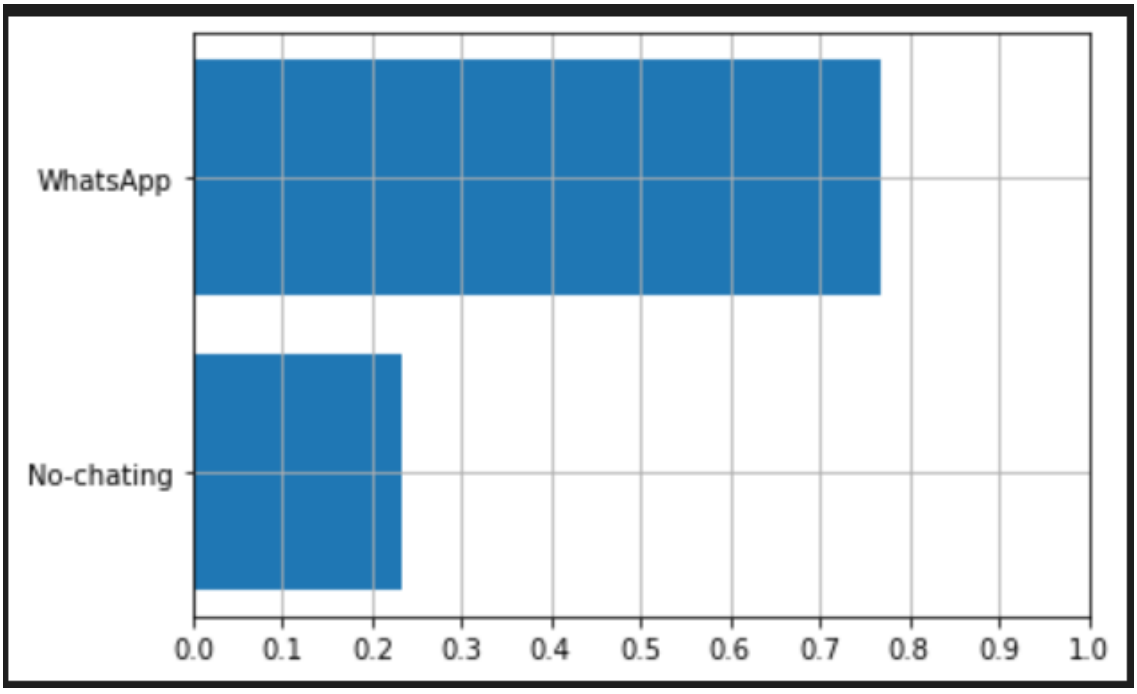


Ilustración 19: Predicción del modelo – predicción 3

Como se puede ver en los resultados anteriores, el modelo es capaz de identificar la ventana de WhatsApp al encontrarse sobre otras ventanas y clasificar la imagen como 'WhatsApp' mostrando su capacidad de detectar herramientas de chat.

4. Imagen de prueba 4

En esta captura se aprecia un escritorio con el explorador de archivos y la aplicación de escritorio de WhatsApp abierta. En ella, las dos aplicaciones se encuentran en modo ventana, estando la de WhatsApp situada parcialmente cubierta por el explorador de archivos.

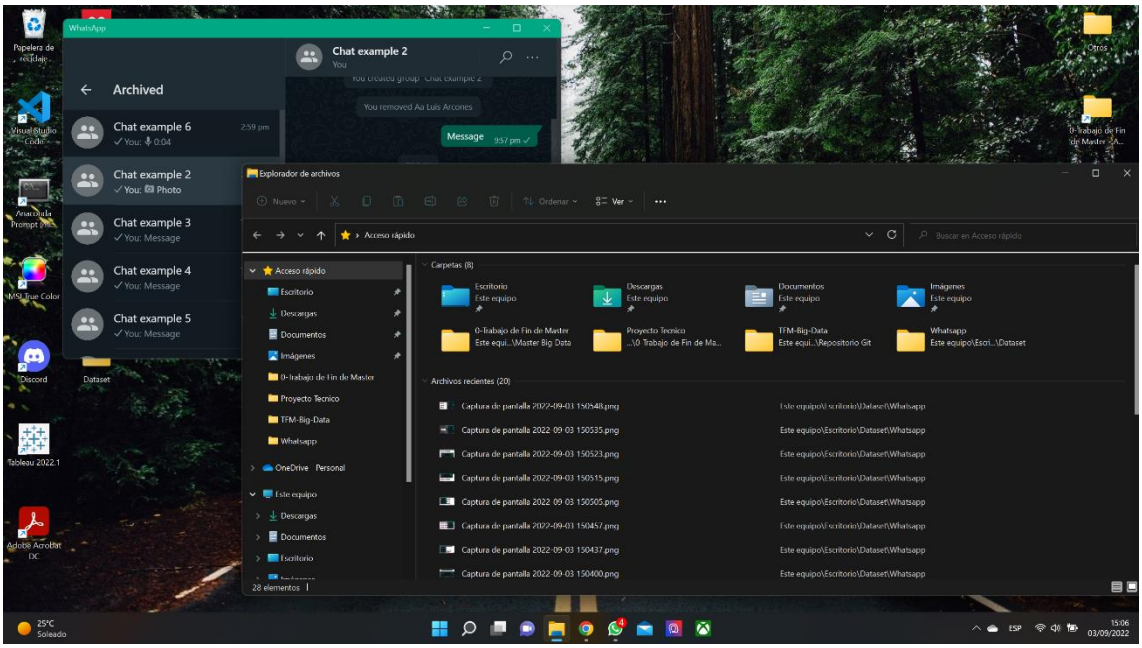


Ilustración 20: Predicción del modelo - imagen 4

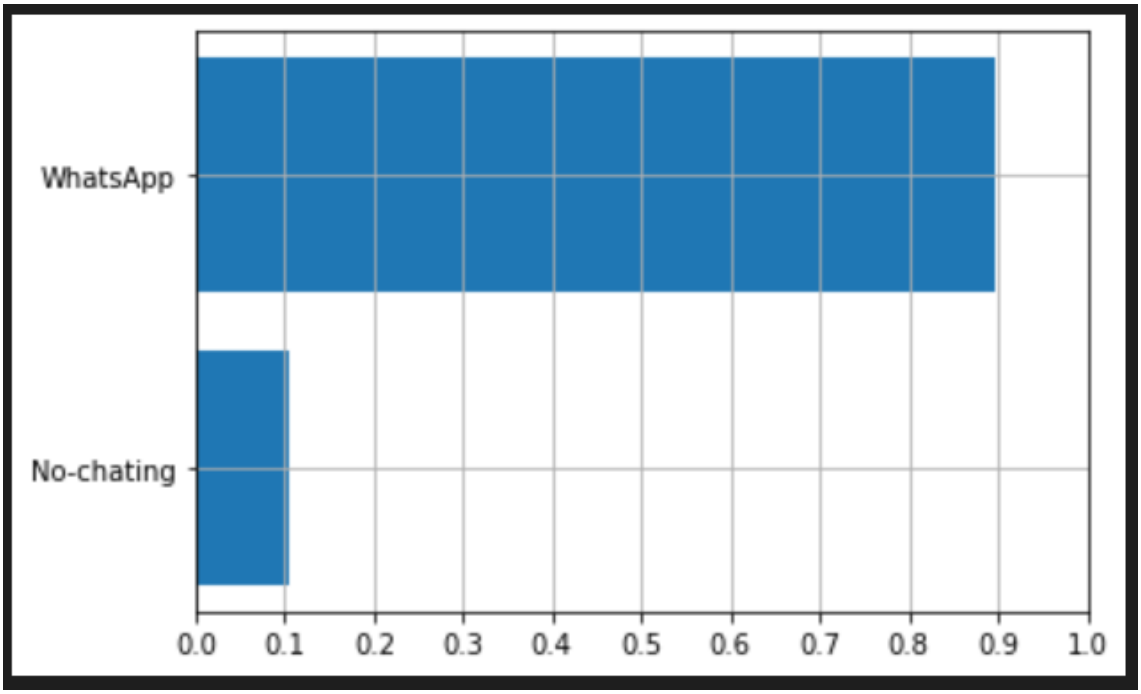


Ilustración 21: Predicción del modelo – predicción 4

Como resultado de la predicción del modelo, la imagen ha sido clasificada como WhatsApp, mostrando un modelo capaz de detectar el uso de este programa incluso cuando este se encuentra parcialmente cubierto por otras ventanas y programas.

Como se puede apreciar en los resultados obtenidos y mostrados anteriormente, el modelo es capaz de identificar y clasificar aquellas capturas de pantalla que poseen la aplicación de WhatsApp abierta, de la misma forma que también es capaz de identificar aquellas que no tienen el programa en ejecución, obteniendo de este proceso un modelo con capacidad para ser usado en sistemas de clasificación e identificación de casos de copia y plagio.

Capítulo 5. Conclusiones y futuras líneas de trabajo

En este apartado se encuentran recogidas las conclusiones extraídas de la realización de este proyecto, así como los diferentes aspectos que pueden suponer una mejora en los resultados y la usabilidad del sistema.

5.1 Conclusiones

5.1.1 Conclusiones del estudio de herramientas

Existen una gran cantidad de herramientas que emplean diferentes tecnologías como el proctoring para garantizar la seguridad y el correcto uso de la tecnología en el entorno académico, previniendo en la medida de lo posible los casos de copia en exámenes presenciales y online.

Sin embargo, las herramientas analizadas basan sus procesos de detección y monitorización de los casos en la presencialidad y la participación activa de los docentes para controlar las conexiones, movimientos sospechosos y grabaciones, lo cual exige una cantidad de tiempo elevada para su uso.

El uso de herramientas automatizadas de big data e inteligencia artificial parece reducir este tiempo, permitiendo a los docentes centrarse en realizar exámenes y formaciones efectivas y reduciendo los casos de forma significativa.

5.1.2 Conclusiones del modelo desarrollado

El modelo desarrollado durante esta práctica ha mostrado una funcionalidad plena que facilita la automatización de la detección del uso de herramientas de chat durante exámenes online.

Sin embargo, algunas conclusiones han sido extraídas en base al data set utilizado, el tamaño de las imágenes empleadas y el modelo construido:

- El data set empleado para el entrenamiento del modelo únicamente poseía 304 imágenes por lo que los resultados obtenidos podrían mejorar en el caso de aumentarlo con imágenes de otros sistemas y escritorios. Eso ha sido debido a los recursos disponibles y a la falta de data sets existentes válidos para este tipo de sistemas.
- El tamaño de las imágenes empleadas condiciona plenamente el funcionamiento del sistema y la cantidad de recursos empleados para su funcionamiento. A medida que el tamaño de las imágenes aumenta, la memoria empleada por el

sistema también lo hace, llegando a suponer un problema para el entrenamiento en equipos convencionales.

- Al tratarse de un data set construido de forma balanceada, únicamente han sido utilizadas las métricas de los y accuracy para entrenar el modelo. Sin embargo, en el caso de aumentarse el data set y obtener clases desbalanceadas otras métricas como la f1, precisión y recall deberían ser estudiadas y aplicadas al modelo.
- El número de epochs utilizadas para entrenar el modelo depende enormemente de los pesos iniciales. De la misma forma, los resultados obtenidos varían en función de dichos pesos. Es por ello, que el uso de la funcionalidad early stoping, aunque facilita limitar el sobre-entrenamiento del modelo, puede suponer que se caiga en mínimos locales.

5.1.3 Conclusiones del sistema implementado con Apache Kafka

El sistema implementado con Apache Kafka y explicado en el apartado 3.5 de este documento ha mostrado ser capaz de enviar, gestionar y procesar en tiempo real numerosos tipos de imágenes de escritorios.

Al implementar productores en lenguaje Python, cualquier sistema que utilice las librerías usadas en este proyecto o similares será capaz de enviar sus capturas al sistema desarrollado y obtener los resultados ofrecidos por el modelo.

Por otra parte, la utilidad de particionado de tópicos ofrecida por la herramienta de Apache Kafka facilita la construcción de un sistema escalable, que a medida que aumente en número de peticiones o de envió de mensajes pueda crecer sin que su rendimiento se vea perjudicado.

Todos los puntos indicados anteriormente se traducen en que el sistema construido con la realización de este proyecto permite de forma sencilla su integración en herramientas ya desarrolladas y su uso en nuevas tecnologías que puedan surgir, proporcionando un sistema fiable que escala en procesamiento a la misma velocidad que aumenta el número de usuarios que lo utilicen.

5.2 Trabajo futuro

Como se puede ver en el apartado 4 de este documento, los resultados de este proyecto han permitido obtener un sistema escalable capaz de predecir en tiempo real cuándo un usuario se encuentra utilizando el programa de WhatsApp. Sin embargo, los resultados obtenidos y la aplicabilidad del sistema podrían mejorar si se aplicasen los siguientes puntos:

- **Aumentar el número de imágenes del data set:** El número de imágenes utilizado para el entrenamiento y predicción del modelo actual es de 304 capturas de pantalla, en las que existen múltiples escenarios contemplados. Sin embargo, existen diferentes combinaciones que no han podido ser recreadas y capturadas, por lo que aumentar el data set con este tipo de imágenes podría mejorar los resultados obtenidos.
- **Aumentar el número de categorías del modelo (Twitter, Facebook):** WhatsApp es una de las redes sociales más utilizadas en la actualidad, sin embargo, no es la única. Con el fin de ofrecer cobertura frente a las otras herramientas de mensajería, se podría ampliar el data set y las clases con capturas de pantalla de escritorios que utilicen estos tipos de tecnología.
- **Aumentar memoria del equipo o trabajar con equipo con más capacidad técnica:** Una de las grandes limitaciones encontradas durante el desarrollo de este proyecto son los recursos empleados para entrenar y procesar el modelo. Al tratarse de imágenes de gran resolución, la memoria del equipo resultó insuficiente en múltiples ocasiones, resultando en una necesidad de reducción de las imágenes para el envío y procesamiento. Aumentando los recursos, también podría aumentar el tamaño de las imágenes y por lo tanto podrían mejorar los resultados obtenidos del modelo.
- **Añadir una interfaz o sistema de visualización de alertas**: Una vez finalizado el proyecto, las imágenes clasificadas por el modelo se almacenan en dos carpetas estáticas, sin posibilidad de identificación del usuario que las envió. Añadir una interfaz visual y una base de datos asociada que permita visualizar y gestionar las alertas podría mejorar significativamente la usabilidad de esta tecnología.

BIBLIOGRAFÍA

[1] Alarcón D.C, "Integridad académica y educación superior: Nuevos retos en la docencia a distancia", Análisis Carolina, 2020

[2] Grande de Prado M., García-Peñalvo F.J, Corell A. y Abella García V., "Evaluación en Educación Superior durante la pandemia de la COVID-19.", Campus Virtuales, 2021.

[3] Hylton K., Levy Y. y Dringus, "Utilizing webcam-based proctoring to deter misconduct in online exams", Computers & Education, 2016.

[4] "Proctoring o como supervisar exámenes online". Grupo Atico34. https://protecciondatos-lopd.com/empresas/proctoring/#Software_para_la_supervision_de_examenes_online.

[5] "Improve the entire examination process". Inspera. https://www.inspera.com/.

[6] "SISTEMA AVANZADO DE PROCTORING, AUTENTIFICACIÓN Y SEGURIDAD QUE PERMITE AUMENTAR LA CONFIANZA EN LA EVALUACIÓN DIGITAL". Mercer. https://www.imercer.com/uploads/common/HTML/LandingPages/mettl/mercer-mettl2.html#Turqbutton.

[7] "The AI-Proctor". AI Proctor. https://ai-proctor.com/.

[8] Razvan Rosu, Alexandru Stefan Stoica, Paul Stefan Popescu University of Craiova y Cristian Marian Mihaescu University of Craiova, "NLP based Deep Learning Approach for Plagiarism Detection", University of Craiova, 2020.

[9] Sara EL KOHLI, Youssef JANNAJ, Mehdi MAANAN y Hassan RHINANE, "DEEP LEARNING: NEW APPROACH FOR DETECTING SCHOLAR EXAMS FRAUD", The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2022.

[10] "SVN vs Git – comparing version control systems". IONOS. https://www.ionos.com/digitalguide/websites/web-development/svn-vs-git-comparing-version-control-systems/.

[11] "Miniconda Vs Anaconda- What Are The Differences?" Developer resources. https://ssiddique.info/miniconda-vs-anaconda.html.

[12] Gastón Caminiti. "¿Qué es y para que sirve Python?" CODERHOUSE. https://legacy.coderhouse.es/blog/que-es-python?utm_term=&utm_campaign=17064052559&utm_source=google_performance_max&utm_medium=cpc&gclid=CjwKCAjwsfuYBhAZEiwA5a6CDKV

NNQO_weGWgaFOpxSkcB0WU5O0Wpt0zkrWYQ_LM5FOk-Ww3FE3mRoCTG4QAvD_BwE.

[13] "TIOBE Index for August 2022". TIOBE. https://www.tiobe.com/tiobe-index/.

[14] "Getting Started". Scikit learn. https://scikit-learn.org/stable/getting_started.html.

[15] Gabriela González. "¿Cuál es la diferencia entre PNG, JPG y GIF?" ThinkBig. https://blogthinkbig.com/diferencia-entre-png-jpg-y-gif.

[16] Espacios de color y el dominio frecuencial. Murcia: Universidad de Murcia.

[17] "Conv2D layer". Keras. https://keras.io/api/layers/convolution_layers/convolution2d/.

[18] Jason Brownlee. "How to Save and Load Your Keras Deep Learning Model". https://machinelearningmastery.com/save-load-keras-deep-learning-models/.

[19] "APACHE KAFKA QUICKSTART". kafka. https://kafka.apache.org/quickstart.

[20] Andrés Plazas. "Leer y escribir datos en Kafka con Python". Medium. https://andres-plazas.medium.com/leer-y-escribir-datos-en-kafka-usando-python-2696154c3948.