



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Grado en Matemáticas e Informática

Trabajo Fin de Grado

**Analizando el Impacto de Modelos
Preentrenados con Deep Learning en
Tareas de Visión Artificial**

Autor: RUBEN GONZALEZ VELASCO
Tutor(a): EMILIO SERRANO FERNANDEZ

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Grado en Matemáticas e Informática

Título: Analizando el Impacto de Modelos Preentrenados con Deep Learning en Tareas de Visión Artificial

Junio 2024

Autor: RUBEN GONZALEZ VELASCO

Tutor: EMILIO SERRANO FERNANDEZ

Departamento de Inteligencia Artificial

Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

Resumen

0.1. Descripción del proyecto

En la última década, la inteligencia artificial y, en particular, las redes neuronales profundas han revolucionado numerosos campos. La capacidad de estos modelos para aprender representaciones complejas de los datos ha permitido avances significativos en tareas como la clasificación de imágenes, encontrando aplicaciones en múltiples industrias. De hecho, el reconocimiento de imágenes y objetos es vital en sectores como la medicina para el diagnóstico asistido, en los coches autónomos para la detección de peatones y señales de tráfico, y en la seguridad para la vigilancia y el reconocimiento facial. También se aplica en realidad aumentada, procesamiento de imágenes médicas y agricultura, donde los drones vigilan los cultivos y detectan plagas. Estas tecnologías permiten automatizar procesos, mejorar la precisión en tareas complejas y facilitar nuevas formas de interacción entre el ser humano y el ordenador.

Sin embargo, el éxito de estos modelos depende a menudo de la disponibilidad de grandes conjuntos de datos y recursos informáticos intensivos, lo que puede suponer un obstáculo importante para muchos investigadores y desarrolladores. Los modelos preformados, entrenados en vastos conjuntos de datos como ImageNet, han surgido como una solución eficaz para superar estas limitaciones. Estos modelos pueden ajustarse a tareas específicas con un esfuerzo de formación relativamente bajo, aprovechando características aprendidas previamente. A pesar de su potencial, la selección del modelo preentrenado más adecuado para una tarea específica sigue siendo un reto, dado el creciente número de arquitecturas y variaciones disponibles.

0.2. Contribución original del proyecto

Este proyecto pretende desarrollar un sistema automatizado que facilite la selección del modelo pre-entrenado más adecuado para tareas específicas de clasificación de imágenes, optimizando así el tiempo y los recursos invertidos en el desarrollo de soluciones basadas en visión por computador. La principal motivación es proporcionar una herramienta que permita a los usuarios identificar rápidamente el modelo pre-entrenado más adecuado a sus necesidades, superando las limitaciones actuales en términos de tiempo y recursos.

0.3. Decisiones de diseño

La selección de Zero-Shot Learning (ZSL) como técnica central de este proyecto se debe a varios factores que subrayan su relevancia y eficacia en el contexto actual de la inteligencia artificial y el aprendizaje profundo:

1. **Versatilidad en la clasificación de nuevas categorías:** ZSL permite clasificar objetos y categorías que no se han visto previamente durante el entrenamiento. Esto es especialmente útil en aplicaciones en las que aparecen constantemente nuevas clases y categorías, como en medicina para identificar nuevas enfermedades o en seguridad para detectar nuevas amenazas.
2. **Reducir la necesidad de datos etiquetados:** Uno de los mayores retos del aprendizaje supervisado tradicional es la necesidad de grandes cantidades de datos etiquetados. ZSL reduce esta dependencia aprovechando las descripciones textuales o los atributos semánticos de las nuevas clases, facilitando así la clasificación sin necesidad de grandes cantidades de datos etiquetados.
3. **Eficiencia de recursos:** Al minimizar la necesidad de entrenamiento con datos nuevos, ZSL permite un uso más eficiente de los recursos computacionales. Esto es crucial para investigadores y desarrolladores con acceso limitado a infraestructuras de alto rendimiento, democratizando el acceso a tecnologías avanzadas.

Para maximizar la accesibilidad y el impacto de esta herramienta, se ha decidido implementarla en una plataforma web. Al estar basada en la web, la aplicación será accesible desde cualquier lugar y en cualquier momento, siempre que se disponga de una conexión a Internet. Esto elimina la necesidad de instalar software especializado en dispositivos locales, permitiendo a los usuarios acceder a la herramienta desde sus ordenadores, tabletas o smartphones.

Este trabajo no busca crear o difundir técnicas ni métodos innovadores. Su objetivo es demostrar que se han adquirido las competencias y resultados de aprendizaje correspondientes a la asignatura "Trabajo Fin de Grado" del "Grado en Matemáticas e Informática

Abstract

0.4. Project Description

In the last decade, artificial intelligence and, in particular, deep neural networks have revolutionized numerous fields. The ability of these models to learn complex representations of data has enabled significant advances in tasks such as image classification, finding applications in multiple industries. Indeed, image and object recognition is vital in sectors such as medicine for assisted diagnosis, in autonomous cars for pedestrian and traffic sign detection, and in security for monitoring and facial recognition. It is also applied in augmented reality, medical image processing and agriculture, where drones monitor crops and detect pests. These technologies enable the automation of processes, improve precision in complex tasks and facilitate new forms of human-computer interaction.

However, the success of these models often depends on the availability of large data sets and intensive computational resources, which can be a significant hurdle for many researchers and developers. Pretrained models, trained on vast datasets such as ImageNet, have emerged as an effective solution to overcome these limitations. These models can be tuned for specific tasks with relatively low training effort, taking advantage of previously learned features. Despite their potential, selecting the most suitable pre-trained model for a specific task remains a challenge, given the growing number of available architectures and variations.

0.5. Original Project Contribution

This project aims to develop an automated system that facilitates the selection of the most suitable pre-trained model for specific image classification tasks, thus optimizing the time and resources invested in the development of computer vision based solutions. The main motivation is to provide a tool that allows users to quickly identify the most suitable pre-trained model for their needs, overcoming current limitations in terms of time and resources.

0.6. Design choices

The selection of Zero-Shot Learning (ZSL) as the central technique in this project is due to several factors that underscore its relevance and effectiveness in the

current context of artificial intelligence and deep learning:

1. **Versatility in Classifying New Categories** ZSL allows the classification of objects and categories that have not been previously seen during training. This is particularly useful in applications where new classes and categories are constantly appearing, such as in medicine for identifying new diseases or in security for detecting new threats.
2. **Reducing the Need for Labeled Data** One of the biggest challenges in traditional supervised learning is the need for large amounts of labeled data. ZSL reduces this dependency by leveraging textual descriptions or semantic attributes of new classes, thus facilitating classification without the need for extensive labeled data.
3. **Resource Efficiency** By minimizing the need for training with new data, ZSL enables more efficient use of computational resources. This is crucial for researchers and developers with limited access to high-performance infrastructure, democratizing access to advanced technologies.

To maximize the accessibility and impact of this tool, it has been decided to implement it on a web platform. Being web-based, the application will be accessible from anywhere at any time, as long as an Internet connection is available. This eliminates the need to install specialized software on local devices, allowing users to access the tool from their computers, tablets or smartphones.

This work does not seek to create or disseminate innovative techniques or methods. Its objective is to demonstrate that the competences and learning outcomes corresponding to the subject “Final Degree Project” of the “Degree in Mathematics and Computer Science” have been acquired.

Tabla de contenidos

0.1. Descripción del proyecto	i
0.2. Contribución original del proyecto	i
0.3. Decisiones de diseño	ii
0.4. Project Description	iii
0.5. Original Project Contribution	iii
0.6. Design choices	iii
1. Introducción	1
2. Desarrollo	3
2.1. PREELIMINARES	3
2.1.1. Aprender Machine Learning desde cero	3
2.1.2. Vision artificial con aprendizaje profundo	3
2.1.3. Modelos pre-entrenados	5
2.1.4. Few-Shot Learning	6
2.1.5. One-Shot Learning	7
2.1.6. Zero-Shot Learning	8
2.2. TRABAJOS RELACIONADOS	10
2.3. CONJUNTOS DE DATOS	10
2.3.1. ImageNet	11
2.3.2. CIFAR-10	11
2.3.3. MNIST	11
2.3.4. Fashion MNIST	12
2.4. MODELOS	12
2.4.1. ResNet50	12
2.4.2. VGG16	12
2.4.3. CLIP-ViT-Large-Patch14	13
2.5. MÉTRICAS	13
2.5.1. Top-1 Accuracy	13
2.5.2. Top-3 Accuracy	13
2.5.3. F1 Score	13
2.6. ENTORNO DE TRABAJO	13
2.6.1. TensorFlow	13
2.6.2. Flask	13
2.6.3. Keras	14
2.6.4. Scikit-learn (sklearn)	14
2.7. METODOLOGIA	14

TABLA DE CONTENIDOS

2.7.1. Índice de objetivos	14
2.7.2. Lista de tareas	14
2.8. IMPLEMENTACIÓN	16
2.8.1. Preproceso de datos	16
2.8.2. Carga de modelos	18
2.8.3. Evaluación	20
2.8.4. Evaluación de Métricas	21
2.9. Aplicaciones Principales	22
2.9.1. Clasificación de Datasets Predeterminados	22
2.9.2. Clasificación con Imágenes Propias	24
2.9.3. Clasificación con Etiquetas Propias	25
2.10. INTERFAZ DE USUARIO	26
2.10.1. Interfaz Web para Clasificación	26
2.10.2. Diseño de la Página	28
2.10.3. Técnicas de Diseño para la Usabilidad	28
3. Análisis de impacto	33
3.0.1. Impacto del Proyecto	33
4. Conclusiones y trabajo futuro	37
4.1. Conclusiones	37
4.1.1. Complejidad Técnica	37
4.1.2. Limitaciones de Tiempo	38
4.1.3. Desafíos en la Integración de Datos	38
4.1.4. Resultados de la Comparación de modelos	38
4.2. Trabajo Futuro	38
4.2.1. Intercambio Sencillo de Modelos, con posible integración de una API como Huggingface	38
4.2.2. Más Métricas	39
4.2.3. Cómputo de Memoria Utilizada	39
4.2.4. Posibilidad de Subir Varias Imágenes o Datasets desde la Web	39
4.2.5. Soporte Multilingüe	39
4.3. Comentario Personal	39
Bibliografía	41
Anexos	47
A. Primer anexo: Índice de originalidad	47

Capítulo 1

Introducción

En el ámbito de la inteligencia artificial y el aprendizaje automático, la visión artificial se ha consolidado como una disciplina fundamental con aplicaciones que van desde la conducción autónoma hasta la medicina. Este proyecto se enfoca en el desarrollo y la implementación de modelos de aprendizaje profundo para la clasificación de imágenes, abordando desde los conceptos preliminares hasta la implementación práctica y el análisis de resultados.

El desarrollo de este proyecto se estructura en varias etapas clave. Comenzamos con una introducción detallada a los conceptos básicos de aprendizaje automático y aprendizaje profundo, incluyendo técnicas avanzadas como Few-Shot, One-Shot y Zero-Shot Learning. Estos métodos permiten la clasificación y reconocimiento de imágenes con cantidades mínimas de datos de entrenamiento, representando un avance significativo en la eficiencia y aplicabilidad de los modelos.

Posteriormente, exploramos trabajos relacionados y revisamos conjuntos de datos ampliamente utilizados en la comunidad de visión artificial, como ImageNet, CIFAR-10, MNIST y Fashion MNIST. Esta revisión proporciona una base sólida para entender los desafíos y avances en el campo.

La sección de modelos presenta un análisis de arquitecturas de redes neuronales populares, como ResNet50, VGG16 y CLIP-ViT-Large-Patch14, evaluando su rendimiento y adecuación para distintas tareas de clasificación. Además, se describen las métricas utilizadas para evaluar estos modelos, como Top-1 Accuracy, Top-3 Accuracy y F1 Score.

En el entorno de trabajo, se detallan las herramientas y frameworks empleados, tales como TensorFlow, Flask, Keras y Scikit-learn, que facilitan el desarrollo, implementación y evaluación de los modelos de aprendizaje profundo.

La metodología del proyecto se estructura en objetivos claros y tareas específicas, garantizando un enfoque organizado y eficiente. La implementación práctica incluye el preproceso de datos, la carga y evaluación de modelos, y la evaluación de métricas, asegurando una aplicación coherente y efectiva de las técnicas desarrolladas.

Capítulo 1. Introducción

Finalmente, se presentan aplicaciones principales, destacando la clasificación de datasets predeterminados y la clasificación con imágenes y etiquetas propias. Se incluye también el desarrollo de una interfaz de usuario intuitiva y funcional, que permite la interacción con los modelos de clasificación a través de una página web diseñada con técnicas de usabilidad.

El análisis de impacto, las conclusiones y el trabajo futuro ofrecen una visión crítica de los logros, desafíos y oportunidades de mejora, estableciendo una base sólida para futuras investigaciones y desarrollos en el campo de la visión artificial y el aprendizaje profundo. Este proyecto no solo contribuye al avance técnico, sino que también plantea un enfoque práctico y accesible para la implementación de modelos de clasificación de imágenes en diversos contextos aplicativos.

Capítulo 2

Desarrollo

2.1. PREELIMINARES

Este trabajo se empieza sin conocimientos previos de aprendizaje profundo, pero sí con una base de matemáticas y programación. Por ello, se ha puesto un especial foco en la revisión del estado del arte.

2.1.1. Aprender Machine Learning desde cero

El objetivo inicial es formar una base de conocimientos sobre Aprendizaje Automático (en inglés Machine Learning) necesaria para conocer el vocabulario usado en los artículos estudiados. Aunque hay una gran variedad de recursos disponibles, esta vez se han utilizado los siguientes:

- Cursos de Introducción al Machine Learning de Kaggle [1]
- Manual de Machine Learning con Scikit-Learn, Keras, y TensorFlow [2]

2.1.2. Vision artificial con aprendizaje profundo

Tras formar una base de conocimientos generales, ponemos el foco en tareas relacionadas con Vision artificial (en inglés Computer Vision). Entre las tareas más populares dentro de esta categoría se encuentran la detección de objetos, la clasificación de Imágenes y la segmentación de imágenes.

La aplicación de técnicas de aprendizaje profundo a estas tareas ganó popularidad gracias a los resultados de AlexNet [3] en el ILSVRC2012 [4].

Redes neuronales de convolución CNN

Una CNN es una arquitectura de modelo de red neuronal que utiliza capas convolucionales. De nuevo, existe gran variedad de recursos que sirven como introducción a este tipo de modelos. Una buena introducción se puede encontrar en el libro online "Visual Guide to Applied Convolution Neural Networks"[5]

Evolución de las CNN El primer ejemplo de CNN "profunda" es LeNet, desarrollado en 1998 por Yann LeCun et. al. [6] la cual fue licenciada a bancos para leer los dígitos de cheques escritos a mano; pero no fue hasta 2012 que otra CNN (AlexNet) tuvo éxito. Entre los modelos que componen el estado del arte se encuentran: la familia de modelos VGG, la cual superó a AlexNet en 2014[7], y la familia de modelos ResNet, dominante en el campo de la visión artificial desde 2015. [8]

El alto coste de entrenamiento de las CNN Las redes neuronales convolucionales (CNN) presentan un alto coste de entrenamiento [9], principalmente debido a varios factores clave:

- **Gran Cantidad de Parámetros:** requiere una cantidad significativa de memoria y poder de cómputo para procesar y ajustar durante el entrenamiento. Por ejemplo, AlexNet, un modelo seminal en CNN, tiene más de 60 millones de parámetros.
- **Varias Capas de Convolución:** Las operaciones convolucionales, que son la base de las CNN, son computacionalmente intensivas. Para obtener buenas clasificaciones, se suelen suceder varias capas de convolución. Cada capa sucesiva extrae características de la capa anterior (y de las características extraídas previamente). Esta cadena de extracciones consecutivas produce características cada vez más "abstractas" que pueden representar mejor las imágenes.

Con el objetivo de evitar estos altos costes de entrenamiento, se pueden aprovechar modelos entrenados previamente, que ofrecen librerías como Keras [10].

La dependencia de datos de los modelos CNN Uno de los problemas comunes en el uso de modelos de aprendizaje profundo, y más concretamente, en el uso de modelos CNN, es que no se disponen de suficientes datos de entrada.

Este problema suele surgir cuando aplicamos aprendizaje profundo a tareas relacionadas con la atención sanitaria, en finanzas, y en comercios para identificar defectos visuales en los productos o para identificar los productos y asignarlos a categorías.

Al disponer de una gran cantidad de parámetros, al entrenar una CNN en un conjunto de datos "pequeño" (menor del orden de 10^5) puede suceder que haya sobreajuste. Es decir, el modelo de aprendizaje automático proporciona predicciones precisas para los datos de entrenamiento, pero no generaliza bien el conocimiento adquirido para los datos nuevos.

En tareas de visión, existen varias estrategias para solucionar este problema:

- **Recolección de datos**
Consiste en obtener más datos para la tarea específica de nuestro interés. La recopilación y preparación de grandes volúmenes de datos para el entrenamiento pueden resultar costosas y consumen mucho tiempo. [11]

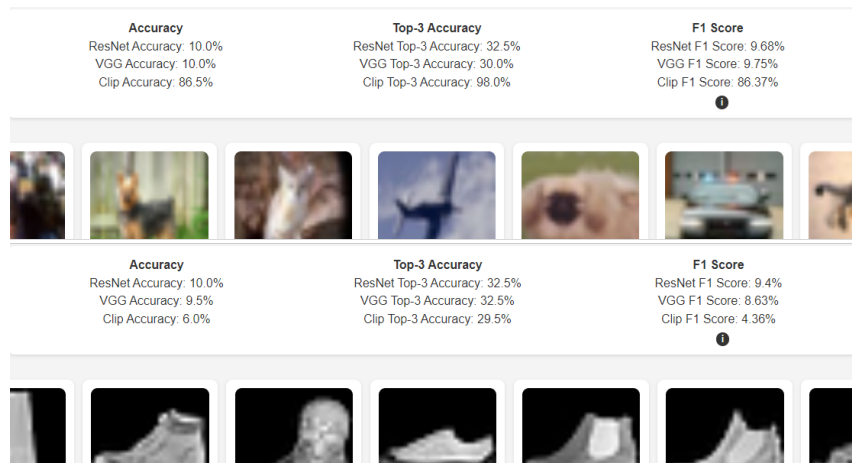


Figura 2.1: Al agregar versiones modificadas de datos ya existentes, podemos aumentar la cantidad de datos. (Imagen de Labeller [13])

- El aumento de datos (data augmentation)

Las técnicas de aumento de datos ayudan a enriquecer los conjuntos de datos al crear muchas variaciones de los datos existentes (figura 2.1). Esto proporciona un conjunto de datos más grande para el entrenamiento y permite que un modelo encuentre características más diversas. Las técnicas de aumento de datos aumentan la eficacia de los conjuntos de datos más pequeños, lo que reduce considerablemente la dependencia de conjuntos de datos de gran tamaño en los entornos de entrenamiento. Además, esta técnica permite obtener un rendimiento mejorado del modelo, a la vez que mejora de la privacidad de los datos, y también una mitigación del sobreajuste en los datos de entrenamiento. [11]

Algunas técnicas de aumento de datos en visión por computadora son Recorte, Volteado, Rotación, Translación, Brillo, Contraste, Aumento de color, y Saturación. [12]

- El uso de modelos preentrenados

Existen varias ventajas al utilizar modelos previamente entrenados, incluida la capacidad de aprovechar el conocimiento y la experiencia de otros. A continuación se trata en mayor profundidad esta técnica.

2.1.3. Modelos pre-entrenados

Un modelo previamente entrenado es un modelo de aprendizaje automático (ML) que se ha entrenado en un gran conjunto de datos y se puede ajustar para una tarea específica, sirviendo como punto de partida para desarrollar modelos de ML.

Este tipo de modelos son populares ya que permiten mejorar significativamente el rendimiento, a la vez que se ahorran tiempo y recursos computacionales.

Los modelos previamente entrenados a menudo han sido entrenados conjuntos

Capítulo 2. Desarrollo

de datos grandes y diversos, para reconocer una amplia gama de patrones y características. Como resultado, pueden proporcionar una base sólida para realizar ajustes. [14] En las tareas de visión artificial, los modelos preentrenados que representan el estado del arte incluyen arquitecturas avanzadas como Res-Net, DenseNet y VGG para la clasificación de imágenes, YOLO, Faster R-CNN y SSD para la detección de objetos, y Mask R-CNN y U-Net para la segmentación. Estos modelos a menudo son entrenados en conjuntos de datos masivos como ImageNet y COCO, y permiten implementar soluciones avanzadas con datos limitados, logrando resultados precisos y eficientes en la clasificación, detección y segmentación de imágenes.

Las dos principales razones por las que utilizar modelos preentrenados son:

- Para ahorrar tiempo y recursos de computación
- Para implementar soluciones avanzadas
- Para reducir la dependencia de datos

Esta introducción a los modelos preentrenados tiene como propósito motivar su uso en situaciones donde se disponen de pocos datos (o ninguno) para representar nuevas clases. Con este fin se desarrollan las técnicas de Zero-shot, One-shot y Few-shot Learning.

Zero-shot, One-shot y Few-shot Learning Las técnicas de Zero-shot, One-shot y Few-shot learning (traducido: aprendizaje de cero, una o pocas oportunidades) son enfoques de aprendizaje que se utilizan para entrenar modelos con muy pocos datos de entrenamiento. [15]

En visión por computadora, una .ºportunidad.^{es} la cantidad de datos que se le proporciona a un modelo para resolver un problema de aprendizaje. En la mayoría de los casos, en aprendizaje profundo para visión por computadora, los modelos reciben grandes cantidades de datos para entrenar, aprender y generar resultados precisos.

N-shot es una escala móvil, desde cero ejemplos (por lo tanto, zero-shot) hasta pocos ejemplos; no más de 5 ejemplos para entrenar un modelo

2.1.4. Few-Shot Learning

(Aprendizaje de pocas posibilidades, FSL): Few-shot learning es una extensión de one-shot learning donde el modelo aprende a partir de un pequeño número de ejemplos (generalmente entre 2 y 100) por clase, Con el objetivo de hacer predicciones o decisiones precisas con solo un puñado de ejemplos. Podemos usar modelos preentrenados aplicando transfer learning.

Transfer Learning

El aprendizaje por transferencia es una técnica de aprendizaje automático en la que un modelo previamente entrenado, que ha sido entrenado en un gran conjunto de datos, se ajusta para una nueva tarea o dominio. [16]

Para tareas de Visión Artificial, son de especial interés las técnicas de Extracción de características (Feature Extraction), y de Ajuste Fino (Fine-Tuning); Aunque existen otras técnicas de aprendizaje por transferencia, como la Adaptación de dominio (domain Adaptation), el Aprendizaje Multi-tarea (Multi-task Learning), y el Aprendizaje Progresivo (Progressive Learning).

Extracción de características (Feature Extraction) Simplemente agregando un nuevo clasificador, que se entrenará desde cero, encima del modelo previamente entrenado el modelo puede reutilizar los mapas de características aprendidos previamente para el conjunto de datos. [17]

Ajuste fino : En esta técnica se descongelan algunas de las capas superiores de un modelo base congelado y se entrenan conjuntamente tanto las capas clasificadoras recién agregadas como las últimas capas del modelo base. Esto nos permite "afinar" las representaciones de características de orden superior en el modelo base para hacerlas más relevantes para la tarea específica. [18]

Aplicaciones :

■ **Reconocimiento de Imágenes y Clasificación:**

- **Aplicaciones Médicas:** Diagnóstico asistido por computadora, análisis de imágenes médicas como radiografías, resonancias magnéticas y tomografías computarizadas.
- **Redes Sociales:** Clasificación automática de imágenes, etiquetado de fotos y detección de contenido inapropiado.
- **Comercio Electrónico:** Búsqueda de productos mediante imágenes, recomendaciones basadas en imágenes.

■ **Reconocimiento Facial:**

- **Seguridad:** Control de acceso basado en reconocimiento facial, identificación de personas buscadas por la policía.
- **Aplicaciones Comerciales:** Personalización de experiencias de compra, análisis de clientes en tiendas físicas.

2.1.5. One-Shot Learning

(Aprendizaje con una oportunidad, OSL): One-shot learning se refiere a la capacidad de un modelo para aprender información suficiente de una sola muestra de una clase nueva para realizar clasificaciones precisas. [19]

Redes neuronales siamesas (SNN) para One-Shot Learning

One-shot learning es un problema de comparación que consiste simplemente en verificar o rechazar si una imagen (o cualquier otra cosa que se presente) coincide con la misma imagen en una base de datos.

Generar esa respuesta implica el uso de una versión de redes neuronales convolucionales (CNN) conocidas como redes neuronales siamesas (SNN). En este tipo de redes, hay dos redes idénticas por las que se pasan dos imágenes idénticas, y después se pasan a través una capa densa con una función de activación sigmoide para producir una puntuación de similitud entre 0 y 1 como función de salida.

Aplicaciones:

Los casos de uso del mundo real implican el reconocimiento y la verificación de rostros y firmas. Como escáneres de pasaportes en aeropuertos o cámaras de vigilancia policial, que buscan terroristas potenciales en lugares públicos concurridos. Los bancos y otras instituciones seguras (incluidos el gobierno, la seguridad, la inteligencia y el ejército) utilizan este enfoque para verificar la identificación o los escaneos biométricos con la copia almacenada en sus bases de datos.

2.1.6. Zero-Shot Learning

(Aprendizaje con cero oportunidades, ZSL): Zero-shot learning permite a un modelo clasificar correctamente ejemplos de clases que nunca ha visto durante el entrenamiento. Esto se logra mediante la utilización de información auxiliar, como descripciones textuales o atributos de las clases.

Visual Transformers

Los Transformers visuales (Visual Transformers) han emergido como una arquitectura poderosa para la clasificación de imágenes, antes dominada por las CNN. A diferencia de las CNN, que operan localmente mediante convoluciones, los Transformers visuales procesan la imagen de manera global, permitiendo captar relaciones a largo plazo entre los píxeles. Un ejemplo destacado de esta arquitectura es el Vision Transformer (ViT), que divide una imagen en parches y los trata como una secuencia de tokens, similar a cómo se procesan las palabras en un Transformer de texto. Esta capacidad de modelar dependencias globales proporciona una mayor flexibilidad y poder de representación, lo que a menudo resulta en un rendimiento superior en tareas de clasificación de imágenes.

CLIP (Contrastive Language-Image Pre-Training)

CLIP, desarrollado por OpenAI, es un innovador modelo que combina la capacidad de los Transformers para el procesamiento de imágenes y textos. CLIP se entrena utilizando un enfoque de preentrenamiento contrastivo, donde se emparejan imágenes y descripciones textuales en grandes volúmenes de datos. Este modelo es capaz de entender relaciones semánticas complejas entre imágenes y texto, lo que le permite realizar tareas de clasificación y búsqueda de imágenes con una notable precisión. Lo que distingue a CLIP es su capacidad para realizar zero-shot learning de manera efectiva, clasificando imágenes en categorías que

no ha visto explícitamente durante el entrenamiento, gracias a su comprensión integrada del lenguaje y las imágenes.

Aplicaciones

■ Realidad Aumentada y Virtual:

- **Entretenimiento:** Integración de objetos virtuales en el mundo real para videojuegos y aplicaciones interactivas.
- **Industria:** Asistencia en reparaciones y mantenimiento mediante superposición de información relevante en el campo de visión del trabajador.

■ Procesamiento de Imágenes Médicas:

- **Diagnóstico y Tratamiento:** Segmentación y análisis de imágenes para identificar tumores, anomalías y planificación de tratamientos.
- **Investigación Médica:** Análisis de grandes volúmenes de datos de imágenes médicas para descubrir patrones y tendencias.

■ Reconocimiento de Texto (OCR):

- **Digitalización de Documentos:** Conversión de imágenes de texto en texto editable para archivos digitales y bases de datos.
- **Automatización de Procesos:** Extracción automática de información de formularios y documentos.

■ Inspección y Control de Calidad:

- **Manufactura:** Inspección automatizada de productos para detectar defectos y asegurar la calidad.
- **Construcción:** Monitoreo del progreso de la construcción y detección de problemas estructurales.

■ Aplicaciones en Agricultura:

- **Monitoreo de Cultivos:** Uso de drones y cámaras para evaluar la salud de los cultivos, detectar enfermedades y optimizar el riego.
- **Gestión del Ganado:** Seguimiento y monitoreo de la salud y comportamiento del ganado.

■ Reconocimiento de Gestos y Movimiento:

- **Interacción Humano-Computadora:** Sistemas de control basados en gestos para dispositivos electrónicos.
- **Salud y Bienestar:** Monitoreo de pacientes para detectar caídas y movimientos anormales.

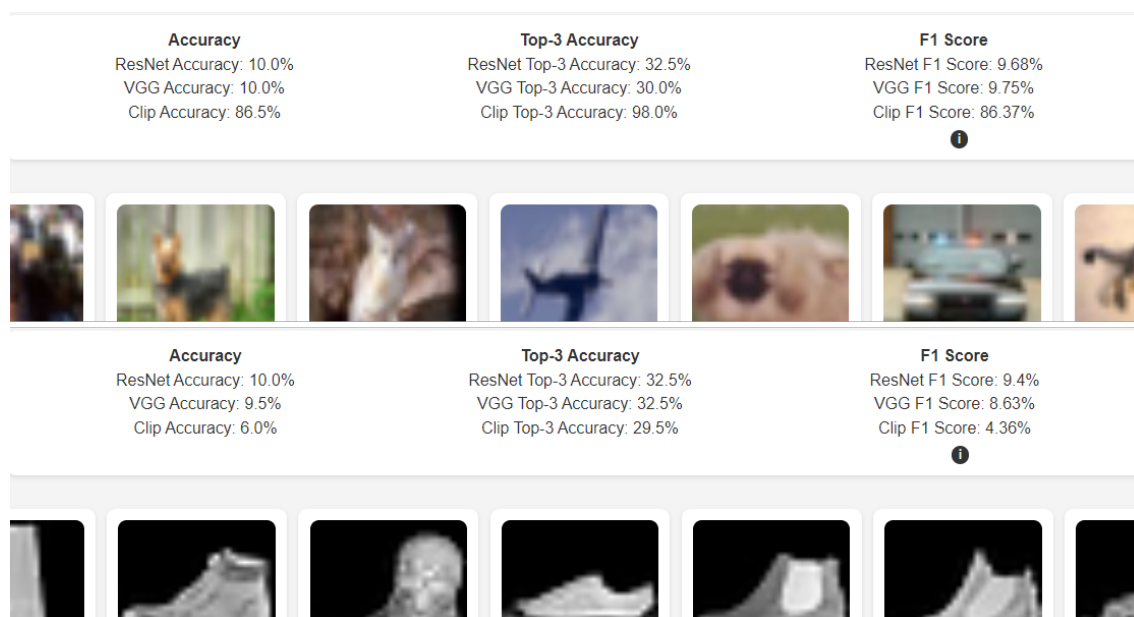


Figura 2.2: resultados de clasificacion en 200 imágenes de CIFAR-10 (Arriba) y en 200 imágenes de Fashion MNIST (abajo)

2.2. TRABAJOS RELACIONADOS

Aplicaciones de Zero-Shot Learning en Clasificación de Imágenes:

Radford et al. (2021), "Learning Transferable Visual Models From Natural Language Supervision", arXiv. [20]

Zero-Shot Learning (ZSL) es una técnica que permite a los modelos clasificar imágenes de clases no vistas durante el entrenamiento. Utiliza representaciones semánticas, como descripciones textuales, para hacer inferencias sobre nuevas clases. Modelos como CLIP (Contrastive Language-Image Pre-training) han revolucionado esta área al permitir la clasificación sin necesidad de entrenamiento específico en las clases objetivo.

2.3. CONJUNTOS DE DATOS

Aunque este trabajo se basa en evitar entrenar modelos, mediante el uso de modelos preentrenados, principalmente en **ImageNet**. Sin embargo, se emplean otros datasets, como CIFAR-10, MNIST y Fashion MNIST, con el objetivo de tener a disposición imágenes etiquetadas con las que hacer pruebas. Además, se pueden observar comportamientos interesantes de la clasificación zero-shot en imágenes en blanco y negro. En efecto, se observa que los resultados de CLIP son muy buenos en CIFAR-10, pero no tanto en Fashion-MNIST. (Figura: 2.2)

2.3.1. ImageNet

ImageNet es una base de datos masiva de imágenes etiquetadas, creada para fomentar y avanzar en el campo de la visión artificial. Esta base de datos contiene millones de imágenes distribuidas en miles de categorías, y ha sido fundamental para entrenar y evaluar algoritmos de aprendizaje profundo. Tanto el modelo ResNet como el modelo VGG utilizado en este trabajo se han entrenado y evaluado extensamente en el conjunto de datos de ImageNet. Además, aunque CLIP no se entrena exclusivamente en ImageNet, utiliza una amplia variedad de datos que pueden incluir imágenes de ImageNet.

```
# Seleccionar el dataset y las etiquetas correspondientes
if dataset == 'cifar10':
    x_test = x_test_cifar10
    y_test = y_test_cifar10
    dataset_labels = cifar10_labels
elif dataset == 'mnist':
    x_test = x_test_mnist
    y_test = y_test_mnist
    dataset_labels = mnist_labels
elif dataset == 'fashion_mnist':
    x_test = x_test_fashion
    y_test = y_test_fashion
    dataset_labels = fashion_labels
else:
    return render_template('select_dataset.html')
```

2.3.2. CIFAR-10

Descripción Técnica: El dataset CIFAR-10 consta de 60,000 imágenes a color de 32x32 píxeles distribuidas en 10 clases diferentes, con 6,000 imágenes por clase. Las clases incluyen aviones, automóviles, aves, gatos, ciervos, perros, ranas, caballos, barcos y camiones. [21]

CIFAR-10 es un benchmark estándar ampliamente utilizado para la evaluación de algoritmos de clasificación de imágenes. La diversidad de las clases y la variabilidad dentro de cada clase lo convierten en un dataset ideal para probar las capacidades de los modelos de Zero-Shot Learning para generalizar a nuevas categorías de imágenes que no fueron vistas durante el entrenamiento.

2.3.3. MNIST

Descripción Técnica: El dataset MNIST contiene 70,000 imágenes en escala de grises de 28x28 píxeles de dígitos manuscritos (del 0 al 9). Está dividido en un conjunto de entrenamiento de 60,000 imágenes y un conjunto de prueba de 10,000 imágenes. [22]

MNIST es uno de los datasets más utilizados para la clasificación de dígitos manuscritos, ofreciendo un entorno controlado y simplificado para probar nuevas

técnicas de aprendizaje automático. Su simplicidad y limpieza lo convierten en un buen punto de partida para evaluar las técnicas de Zero-Shot Learning en tareas de reconocimiento de patrones básicos antes de aplicarlas a datasets más complejos.

2.3.4. Fashion MNIST

Descripción Técnica: Fashion MNIST es un dataset que contiene 70,000 imágenes en escala de grises de 28x28 píxeles de artículos de moda, como camisas, pantalones, vestidos y zapatos, distribuidos en 10 categorías. Al igual que MNIST, se divide en un conjunto de entrenamiento de 60,000 imágenes y un conjunto de prueba de 10,000 imágenes. [23]

Fashion MNIST fue creado como un reemplazo más desafiante para MNIST, con imágenes que son más complejas y con más variación intraclase. Esto lo hace adecuado para probar modelos de Zero-Shot Learning, ya que estos modelos deben ser capaces de capturar características más abstractas y semánticas para realizar una clasificación efectiva en nuevas categorías de productos de moda.

2.4. MODELOS

En esta sección se presentan breves descripciones de los modelos utilizados en el proyecto para la clasificación de imágenes.

2.4.1. ResNet50

Descripción: ResNet50 es una red neuronal convolucional de 50 capas diseñada para facilitar el entrenamiento de redes neuronales más profundas. Introduce un esquema de aprendizaje residual que permite el uso de "shortcuts,"^o conexiones de salto entre capas, mitigando el problema del desvanecimiento del gradiente y mejorando la precisión del modelo. ResNet50 ha sido preentrenado en el dataset ImageNet, proporcionando una base sólida para tareas de transferencia de aprendizaje.

- *He et al. (2016)*, "Deep Residual Learning for Image Recognition", CVPR. [8]

2.4.2. VGG16

Descripción: VGG16 es una red neuronal convolucional profunda con 16 capas de peso, desarrollada por el Visual Geometry Group (VGG) de la Universidad de Oxford. Su arquitectura se caracteriza por tener muchas capas convolucionales pequeñas (3x3) seguidas de capas de agrupamiento (pooling). VGG16 también ha sido preentrenada en el dataset ImageNet, lo que la hace útil para tareas de transferencia de aprendizaje.

- *Simonyan y Zisserman (2015)*, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR. [7]

2.4.3. CLIP-ViT-Large-Patch14

Descripción: CLIP (Contrastive Language-Image Pre-Training) es un modelo desarrollado por OpenAI que utiliza una combinación de visión y lenguaje natural para realizar tareas de clasificación de imágenes. La variante ViT-Large-Patch14 se basa en un Transformer de gran tamaño para la parte de visión, utilizando patches de 14x14 píxeles. CLIP está entrenado para asociar textos e imágenes, permitiendo la clasificación de imágenes mediante descripciones textuales, lo cual es especialmente útil en escenarios de Zero-Shot Learning.

- Radford et al. (2021), "Learning Transferable Visual Models From Natural Language Supervision", arXiv. [20]

2.5. MÉTRICAS

2.5.1. Top-1 Accuracy

La precisión Top-1 mide la proporción de veces que la clase con la probabilidad más alta coincide con la clase real de la imagen. Evalúa la capacidad del modelo para identificar correctamente nuevas clases no vistas durante el entrenamiento.

2.5.2. Top-3 Accuracy

La precisión Top-3 mide la proporción de veces que la clase real de la imagen se encuentra entre las tres clases con mayor probabilidad. Proporciona una evaluación más flexible del rendimiento del modelo, útil en casos donde la exactitud perfecta no es crítica.

2.5.3. F1 Score

La puntuación F1 es la media armónica de la precisión y la sensibilidad, considerando tanto falsos positivos como falsos negativos. Ofrece una medida equilibrada del rendimiento del modelo en conjuntos de datos desbalanceados y en tareas de clasificación con múltiples clases.

2.6. ENTORNO DE TRABAJO

2.6.1. TensorFlow

Descripción: TensorFlow es una biblioteca de código abierto desarrollada por Google para el aprendizaje automático. Permite la creación y entrenamiento de modelos de aprendizaje profundo y es compatible con una variedad de plataformas y dispositivos.

2.6.2. Flask

Descripción: Flask es un microframework de Python para el desarrollo de aplicaciones web. Es ligero y flexible, facilitando la creación rápida de aplicaciones

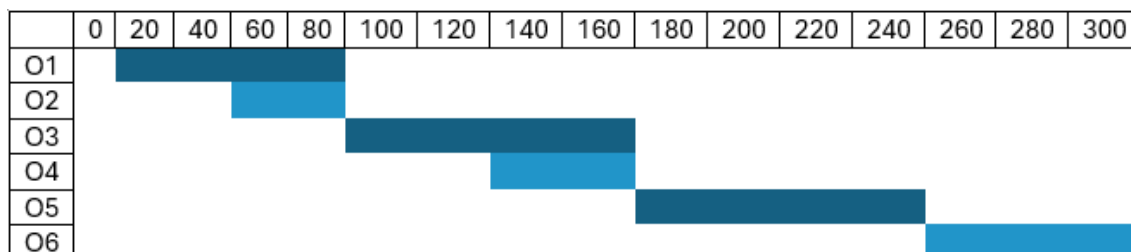


Figura 2.3: Diagrama de Gantt del Proyecto

web y API.

2.6.3. Keras

Descripción: Keras es una API de alto nivel para redes neuronales que se ejecuta sobre TensorFlow. Proporciona una interfaz simple y modular para crear y entrenar modelos de aprendizaje profundo.

2.6.4. Scikit-learn (sklearn)

Descripción: Scikit-learn es una biblioteca de Python para el aprendizaje automático. Ofrece herramientas eficientes para el modelado y análisis de datos, incluyendo algoritmos de clasificación, regresión y clustering.

2.7. METODOLOGIA

En este proyecto se ha seguido una metodología ágil, siguiendo el diagrama de Gantt de la figura 2.3 aplicando una entrega incremental y continua de funcionalidades, y adaptándose a los cambios, aun en etapas avanzadas del desarrollo.

2.7.1. Índice de objetivos

- O1. Revisión Bibliográfica y Análisis del Estado del Arte
- O2. Estudio del Impacto de Modelos Preentrenados
- O3. Desarrollo de un Sistema de Selección de Modelos
- O4. Evaluación del Sistema Propuesto
- O5. Implementación del Sistema de Selección de Modelos en un entorno web.
- O6. Documentación y Presentación

2.7.2. Lista de tareas

O1. Revisión Bibliográfica y Análisis del Estado del Arte:

- Investigar y analizar la literatura existente sobre visión por computador y redes neuronales simples.

- Identificar propuestas y técnicas relevantes del estado del arte.
- Evaluar los puntos fuertes y débiles de cada propuesta.

O2. Estudio del Impacto de Modelos Preentrenados

- Comparar el rendimiento de modelos preentrenados con respecto a los enfoques tradicionales en diferentes conjuntos de datos.
- Considerar diferentes arquitecturas de modelos preentrenados y evaluar su eficacia en diversas condiciones.

O3. Desarrollo de un Sistema de Selección de Modelos:

- Diseñar y desarrollar un sistema que pueda identificar automáticamente el modelo preentrenado más adecuado para una tarea específica y un conjunto de datos determinado.
- Implementar algoritmos de selección que consideren métricas de rendimiento, tipo de tarea y características del conjunto de datos.

O4. Evaluación del Sistema Propuesto:

- Evaluar el sistema de selección de modelos desarrollado en términos de precisión y eficacia.
- Comparar los resultados obtenidos por el sistema con la elección manual de modelos preentrenados para validar su utilidad y eficiencia.

O5. Implementación del Sistema de Selección de Modelos en un entorno web.

- Diseñar una interfaz de usuario intuitiva y fácil de usar para el sistema de selección de modelos.
- Implementar la lógica de selección de modelos en el backend, garantizando un rendimiento eficiente.
- Proporcionar información detallada sobre el rendimiento de los modelos seleccionados.

O6. Documentación y Presentación:

- Elaborar una documentación completa que incluya la metodología, resultados y conclusiones del trabajo.
- Documentar el proceso de desarrollo web, detallando las tecnologías utilizadas, decisiones de diseño y la arquitectura general del sistema.

2.8. IMPLEMENTACIÓN

Implementación (con detalles sobre la solución/implementación realizada).

2.8.1. Preproceso de datos

El preprocesamiento de datos es una parte fundamental del flujo de trabajo de aprendizaje automático. En este código, se realizan varias tareas de preprocesamiento para preparar los datos antes de entrenar o evaluar los modelos. Estas tareas incluyen:

- Carga de conjuntos de datos predeterminados
- Modificación de las dimensiones de las imágenes
- Ajuste del tamaño de las imágenes
- Extracción de etiquetas

Carga de conjuntos de datos predeterminados

El código carga varios conjuntos de datos predeterminados que se utilizan comúnmente en tareas de clasificación de imágenes: CIFAR-10, MNIST y Fashion MNIST. Para cada conjunto de datos, se selecciona un subconjunto pequeño para ahorrar memoria al cargar la aplicación.

Listing 2.1: Carga de datos

```
# Load and preprocess CIFAR-10 dataset
(x_train_cifar10, y_train_cifar10), (x_test_cifar10, y_test_cifar10) =
load_and_preprocess_dataset(cifar10, subset_size=200, expand_dims=False,
repeat_channels=False)
cifar10_labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
'frog', 'horse', 'ship', 'truck']

# Load and preprocess MNIST dataset
(x_train_mnist, y_train_mnist), (x_test_mnist, y_test_mnist) =
load_and_preprocess_dataset(mnist, subset_size=200, expand_dims=True,
repeat_channels=True)
mnist_labels = [str(i) for i in range(10)]

# Load and preprocess Fashion MNIST dataset
(x_train_fashion, y_train_fashion), (x_test_fashion, y_test_fashion) =
load_and_preprocess_dataset(fashion_mnist, subset_size=200,
expand_dims=True, repeat_channels=True)
fashion_labels = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle_boot']
```


Modificación de las dimensiones de las imágenes

Para los conjuntos de datos MNIST y Fashion MNIST, las imágenes originales son en escala de grises (un canal). Sin embargo, muchos modelos preentrenados, como ResNet50 y VGG16, esperan entradas en color (tres canales). Para solucionar esto, se expanden las dimensiones de las imágenes para incluir un canal adicional y luego se repiten los canales para crear una imagen RGB.

Listing 2.2: Preproceso de datos

```
# Si las imágenes están en blanco y negro, hay que expandirlas
# para que tengan las mismas dimensiones que el formato RGB
def load_and_preprocess_dataset(dataset_loader, subset_size=200,
                                expand_dims=False, repeat_channels=False, n_channels=3):
    (x_train, y_train), (x_test, y_test) = dataset_loader.load_data()
    x_test, y_test = x_test[:subset_size], y_test[:subset_size]

    if expand_dims:
        x_train = np.expand_dims(x_train, axis=-1)
        x_test = np.expand_dims(x_test, axis=-1)

    if repeat_channels:
        x_train = np.repeat(x_train, n_channels, axis=-1)
        x_test = np.repeat(x_test, n_channels, axis=-1)

    return (x_train, y_train), (x_test, y_test)
```

Ajuste del tamaño de las imágenes

Se usa una función de la librería numpy para redimensionar las imágenes al tamaño específico de entrada que requieren los modelos:

Listing 2.3: Ajuste del tamaño de las imágenes

```
def resize_image(image, target_size=(32, 32)):
    return tf.image.resize(image, target_size).numpy()
```

Extracción de etiquetas

Para el conjunto de datos ImageNet, del cual solo nos interesan las etiquetas de las clases, se descarga un archivo JSON que contiene el índice de clases y sus etiquetas correspondientes. Este archivo se procesa para extraer las etiquetas y se guardan en un archivo de texto para su uso posterior.

Listing 2.4: Carga de etiquetas a un archivo

```
# URL to fetch the ImageNet class index to label mapping
url = 'https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json'
response = requests.get(url)
class_idx = response.json()
```

Capítulo 2. Desarrollo

```
# Extract labels from the class index dictionary
imagenet_labels = [class_idx[str(i)][1] for i in range(len(class_idx))]
# Save the labels to imagenet_labels.txt
with open('imagenet_labels.txt', 'w') as f:
    for label in imagenet_labels:
        f.write(f"{label}\n")
logging.info("ImageNet_labels_have_been_saved_to_imagenet_labels.txt")
```

Conversión de imágenes a base 64

Al finalizar este proceso, la variable `img_str` contendrá la representación de la imagen en formato base64, lista para ser utilizada en un entorno HTML. Para ello, se carga la imagen original en formato PIL (Python Imaging Library) desde un array numpy. Se guarda la imagen en un buffer utilizando BytesIO y se especifica el formato de salida como PNG. Finalmente, se convierte la imagen en el buffer a formato base64 y se decodifica para obtener una cadena de texto.

Listing 2.5: Carga de etiquetas a un archivo

```
# Convertir imagen a formato base64 para mostrar en HTML
img_pil = Image.fromarray(test_image.astype('uint8'))
buffered = BytesIO()
img_pil.save(buffered, format="PNG")
img_str = base64.b64encode(buffered.getvalue()).decode()
```

2.8.2. Carga de modelos

La carga de modelos se realiza de manera dinámica según sea necesario. Los modelos utilizados son ResNet50 y VGG16, ambos disponibles en Keras con pesos preentrenados en ImageNet. También se incluye el modelo CLIP de OpenAI para clasificación basada en texto.

Carga de ResNet50 y VGG16

Para cargar los modelos ResNet50 y VGG16, se utiliza la técnica de **transfer learning**. Esta técnica permite utilizar un modelo preentrenado en un conjunto de datos grande (como ImageNet) y adaptarlo para una nueva tarea de clasificación. Los pasos seguidos para la carga de cada modelo son:

- **Carga del modelo base:** Se carga el modelo preentrenado sin la capa superior (`include_top=False`) utilizando los pesos entrenados en el conjunto de datos ImageNet.
- **Agregación de capas adicionales:** Sobre la salida del modelo base, se añade una capa de `GlobalAveragePooling2D` para reducir la dimensionalidad y una capa `Dense` con activación `softmax` para la clasificación en 10 categorías. Es importante destacar que estas capas adicionales no se entrenan, manteniendo los pesos del modelo base fijos. Esta decisión se toma para

reducir el tiempo de entrenamiento y aprovechar directamente los conocimientos previamente adquiridos por el modelo en el conjunto de datos ImageNet. Al no entrenar las capas adicionales, se limita la capacidad del modelo para adaptarse a la nueva tarea, y por tanto se obtendrá una menor precisión a la hora de clasificar.

- **Creación del modelo final:** Se crea el modelo final definiendo las entradas del modelo base y las salidas de las capas adicionales agregadas.

Listing 2.6: Carga de ResNet50 y VGG16

```
def load_model(model_cls, preprocess_input):
    base_model = model_cls(weights='imagenet', include_top=False)
    x = GlobalAveragePooling2D()(base_model.output)
    predictions = Dense(10, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    return model, preprocess_input
```

Carga del modelo CLIP

El modelo CLIP de OpenAI se carga una sola vez al inicio del programa y se utiliza para la clasificación basada en texto. La técnica utilizada aquí se conoce como **zero-shot learning**. Los pasos para la carga del modelo CLIP son:

- **Carga del modelo y procesador:** Se utiliza la clase `CLIPModel` para cargar el modelo CLIP preentrenado (`clip-vit-base-patch32`) y la clase `CLIPProcessor` para manejar el preprocesamiento de entrada.
- **Clasificación con CLIP:** La función `clip_classify` se encarga de clasificar imágenes utilizando el modelo CLIP. Convierte la imagen a un formato adecuado, la procesa junto con las etiquetas de texto, y calcula las probabilidades de clasificación.

La técnica de **zero-shot learning** permite que el modelo CLIP clasifique imágenes en categorías para las cuales no ha sido específicamente entrenado, utilizando descripciones textuales como referencia.

Listing 2.7: Carga del modelo CLIP

```
# Load CLIP model once at the start
clip_model =
CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
clip_processor =
CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")

def clip_classify(image_array, text_labels):
    image = Image.fromarray(image_array.astype('uint8'))
    inputs = clip_processor(text=text_labels, images=image,
return_tensors="pt", padding=True)
    start_time = time.time()
    outputs = clip_model(**inputs)
```

```
end_time = time.time()
logits_per_image = outputs.logits_per_image
probs = logits_per_image.softmax(dim=1).tolist()
elapsed_time = end_time - start_time
return probs, elapsed_time
```

En resumen, la carga de los modelos ResNet50 y VGG16 se realiza mediante **transfer learning**, agregando capas adicionales que no se entrenan para aprovechar los pesos preentrenados. El modelo CLIP utiliza **zero-shot learning** para clasificar imágenes en categorías basadas en descripciones textuales, permitiendo una clasificación flexible sin entrenamiento específico en esas categorías.

2.8.3. Evaluación

Clasificación de Imágenes (con Resnet y VGG)

La función para clasificar imágenes recibe la imagen, el modelo y la función de preprocesamiento, y devuelve las predicciones y el tiempo de procesamiento:

Listing 2.8: Funcion de Clasificación con ResNet y Vgg

```
def classify_image(image_array, model_cls, preprocess_input):
    # Cargar el modelo base y la función de preprocesamiento
    model, preprocess_input = load_model(model_cls, preprocess_input)
    # Redimensionar la imagen al tamaño requerido por el modelo
    image_resized = resize_image(image_array)
    # Preprocesar la imagen redimensionada
    processed_img = preprocess_input(np.expand_dims(image_resized, axis=0))
    # Registrar el tiempo de inicio antes de realizar la predicción
    start_time = time.time()
    # Realizar la predicción con el modelo
    predictions = model.predict(processed_img)
    # Registrar el tiempo después de realizar la predicción
    end_time = time.time()
    # Calcular el tiempo transcurrido durante la predicción
    elapsed_time = end_time - start_time
    # Devolver las predicciones y el tiempo transcurrido
    return predictions, elapsed_time
```

Clasificación con el Modelo CLIP

El modelo CLIP se utiliza para clasificar imágenes basándose en una lista de etiquetas de texto. Se usa la misma función de clasificación vista anteriormente:

Listing 2.9: Clasificación con el Modelo CLIP

```
def clip_classify(image_array, text_labels):
    # Convertir el array de la imagen a un objeto de imagen PIL
    image = Image.fromarray(image_array.astype('uint8'))
    # Preparar las entradas para el modelo CLIP con las etiquetas
```

```
# de texto y la imagen
inputs = clip_processor(text=text_labels, images=image,
return_tensors="pt", padding=True)
# Registrar el tiempo de inicio antes de realizar la predicción
start_time = time.time()
# Realizar la predicción con el modelo CLIP
outputs = clip_model(**inputs)
# Registrar el tiempo después de realizar la predicción
end_time = time.time()
# Obtener los logits de la predicción por imagen
logits_per_image = outputs.logits_per_image
# Convertir los logits a probabilidades usando la función softmax
probs = logits_per_image.softmax(dim=1).tolist()
# Calcular el tiempo transcurrido durante la predicción
elapsed_time = end_time - start_time
# Devolver las probabilidades y el tiempo transcurrido
return probs, elapsed_time
```

Obtener las Mejores Predicciones

Para mostrar las 3 mejores predicciones, se seleccionan las *top-k* predicciones con mayor probabilidad:

```
def get_top_predictions(predictions, top_k=3):
    class_indices = np.argsort(predictions[0])[:,::-1][:top_k]
    return [(i, predictions[0][i]) for i in class_indices]
```

2.8.4. Evaluación de Métricas

Precisión

Listing 2.10: Cálculo de precisión

```
# Seleccionar una imagen aleatoria y su índice
test_image = x_test[idx]
actual_label_index =
y_test[idx][0] if dataset == 'cifar10' else y_test[idx]
actual_label = dataset_labels[actual_label_index]

# Clasificar la imagen usando ResNet50 y registrar el tiempo.
resnet_predictions, resnet_time =
classify_image(test_image, ResNet50, preprocess_input_resnet)
resnet_top_preds = get_top_predictions(resnet_predictions)

# Verificar si la predicción principal es correcta.
resnet_is_correct = resnet_top_preds[0][0] == actual_label_index
if resnet_is_correct:
# Incrementar el contador de predicciones correctas si hay acierto
resnet_correct_predictions += 1
```

Capítulo 2. Desarrollo

```
# Calcular la precisión como porcentaje.
resnet_accuracy = (resnet_correct_predictions / num_images) * 100
```

Precisión Top-3

```
# Compute top3-accuracy
# Si la etiqueta correcta está entre las 3 predicciones principales,
resnet_top3_correct =
actual_label_index in [pred[0] for pred in resnet_top_preds]
# Incrementar el contador de predicciones top-3 correctas.
if resnet_top3_correct:
    resnet_top3_correct_predictions += 1
# Calcular la precisión top-3 como porcentaje.
resnet_top3_accuracy =
(resnet_top3_correct_predictions / num_images) * 100
```

Otras métricas (F1 score)

El puntaje F1 es la media armónica de la precisión y la exhaustividad (recall). Para calcularlo vamos a utilizar la librería sklearn, donde si fuese necesario podríamos implementar fácilmente otras métricas como 'top_k_accuracy' o 'jaccard_score'.

```
from sklearn.metrics import f1_score
# Compute f1-score
# Inicializar listas para etiquetas verdaderas y predichas
y_true = []
resnet_y_pred = []
for idx in indices:
    # Agregar la etiqueta verdadera a la lista.
    y_true.append(actual_label_index)
    resnet_top_preds = get_top_predictions(resnet_predictions)
    # Agregar la predicción principal a la lista de predicciones.
    resnet_y_pred.append(resnet_top_preds[0][0])
    # Calcular el puntaje F1 macro
    resnet_f1 = f1_score(y_true, resnet_y_pred, average='macro')
```

En resumen, las funciones de evaluación permiten cargar modelos, procesar imágenes, y realizar clasificaciones.

2.9. Aplicaciones Principales

2.9.1. Clasificación de Datasets Predeterminados

Esta aplicación permite clasificar imágenes utilizando datasets predeterminados como CIFAR-10, MNIST y Fashion-MNIST. A continuación se explica paso a paso el proceso y las técnicas utilizadas para la clasificación.

1. Carga de Datasets

2. Preparación de los Modelos

3. Clasificación:

El usuario selecciona el dataset y el número de imágenes a clasificar. Se realiza la clasificación utilizando los modelos preparados.

4. Calculo Métricas de Evaluación:

Se evalúa el rendimiento calculando la precisión y otras métricas.

Listing 2.11: Aplicación principal: Clasificación y Evaluación

```
@app.route('/classify', methods=['GET', 'POST'])
def classify():
    global classification_active
    logging.debug("Processing_dataset_classification")

    # Almacena datos seleccionados desde la plantilla HTML
    dataset = request.form['dataset']
    num_images = int(request.form['number'])

    # Seleccionar imágenes del dataset ...
    # Inicializar variables para el cálculo de métricas...

    # Selección aleatoria de imágenes del dataset
    indices = np.random.choice(len(x_test), num_images, replace=False)

    # Detiene la clasificación cuando el usuario quiera
    classification_active = True

    for idx in indices:
        if not classification_active:
            break

        # Prepara la imagen seleccionada y su etiqueta asociada
        test_image = x_test[idx]
        actual_label_index = y_test[idx][0] if dataset == 'cifar10'
        else y_test[idx]
        actual_label = dataset_labels[actual_label_index]

        # Clasificación de imagen
        resnet_predictions, resnet_time =
            classify_image(test_image, ResNet50, preprocess_input_resnet)
        vgg_predictions, vgg_time =
            classify_image(test_image, VGG16, preprocess_input_vgg)
        clip_probs, clip_time =
            clip_classify(test_image, labels)

    # Convertir imagen a formato base64 para mostrar en HTML ...
```

```
# Prepara results para mandarlo a la plantilla HTML
results.append({
    'image_data': img_str,
    'actual_label': actual_label,
    'resnet_predictions': resnet_top_preds,
    'vgg_predictions': vgg_top_preds,
    'clip_predictions': clip_top_preds,
    'resnet_correct': resnet_is_correct,
    'vgg_correct': vgg_is_correct,
    'clip_correct': clip_is_correct,
    'resnet_time': resnet_time,
    'vgg_time': vgg_time,
    'clip_time': clip_time,
})

# manda results a la plantilla HTML correspondiente
return render_template('classification_results.html',
    labels=dataset_labels,
    results=results,
    resnet_accuracy=resnet_accuracy,
    vgg_accuracy=vgg_accuracy,
    resnet_top3_accuracy=resnet_top3_accuracy,
    vgg_top3_accuracy=vgg_top3_accuracy,
    resnet_f1=resnet_f1,
    vgg_f1=vgg_f1)
```

2.9.2. Clasificación con Imágenes Propias

El usuario puede cargar una imagen propia, que es procesada y clasificada utilizando los modelos preentrenados ResNet50, VGG16 y CLIP. Los resultados de las predicciones se muestran junto con el tiempo de clasificación.

Listing 2.12: Aplicación: Clasificación de imágenes propias

```
if file:
    # Código para la clasificación de imágenes propias
    logging.debug("Processing_uploaded_image")

    # Convertir la imagen a un formato manejable
    image = Image.open(file.stream)
    image = image.convert('RGB') # Ensure RGB format
    image_array = np.array(image)

    # Procesa y clasifica la imagen
    resnet_predictions, resnet_time =
    classify_image(image_array, ResNet50, preprocess_input_resnet)
    vgg_predictions, vgg_time =
```



```
classify_image(image_array, VGG16, preprocess_input_vgg)
clip_probs, clip_time = clip_classify(image_array, labels)

# Obtener las 3 mejores predicciones ...

# Mapear las predicciones con sus etiquetas correspondientes
resnet_top_preds =
[(labels[idx], prob) for idx, prob in resnet_top_preds]
vgg_top_preds =
[(labels[idx], prob) for idx, prob in vgg_top_preds]
clip_top_preds =
[(labels[idx], prob) for idx, prob in clip_top_preds]

# Convertir imagen a formato base64 para mostrar en HTML ...

# Preparar result para pasarlo a la plantilla HTML
result = {
    'image_data': img_str,
    'resnet_predictions': resnet_top_preds,
    'vgg_predictions': vgg_top_preds,
    'clip_predictions': clip_top_preds,
    'resnet_time': resnet_time,
    'vgg_time': vgg_time,
    'clip_time': clip_time,
}
# Pasar result a la plantilla correspondiente
return render_template('upload_result.html', result=result)
```

2.9.3. Clasificación con Etiquetas Propias

El usuario puede definir etiquetas personalizadas para la clasificación de imágenes. Este método utiliza el modelo CLIP para asociar las etiquetas personalizadas con la imagen cargada.

Listing 2.13: Aplicación: Clasificación de imágenes con etiquetas propias

```
@app.route('/classify_custom', methods=['GET', 'POST'])
def classify_custom():
    # Código para la clasificación de imágenes propias
    logging.debug("POST_request_for_custom_label_classification")
    # Almacena datos seleccionados desde la plantilla HTML
    file = request.files.get('file_custom')
    custom_labels_only = request.form.get('custom_labels_only')
    if file and custom_labels_only:
        # Procesa y clasifica la imagen
        labels =
        [label.strip() for label in custom_labels_only.split(',')]
        # Convertir la imagen a un formato manejable
```

Capítulo 2. Desarrollo

```
image = Image.open(file.stream)
image = image.convert('RGB')
image_array = np.array(image)

clip_probs, clip_time = clip_classify(image_array, labels)
# Mapear las 3 mejores predicciones con sus etiquetas
clip_top_preds =
get_top_predictions(np.array([clip_probs[0]]))
clip_top_preds =
[(labels[idx], prob) for idx, prob in clip_top_preds]
# Convertir imagen a formato base64 para mostrar en HTML
buffered = BytesIO()
image.save(buffered, format="PNG")
img_str = base64.b64encode(buffered.getvalue()).decode()
# Prepara results para mandarlo a la plantilla HTML
result = {
    'image_data': img_str,
    'clip_predictions': clip_top_preds,
    'clip_time': clip_time,
}
# manda results a la plantilla HTML correspondiente
return render_template('classify_custom.html', result=result)
```

2.10. INTERFAZ DE USUARIO

2.10.1. Interfaz Web para Clasificación

Se define la interfaz web usando Flask, permitiendo a los usuarios cargar imágenes y seleccionar datasets para clasificación.

Descripción Breve de Cada Plantilla Utilizada

select_dataset.html Esta plantilla proporciona la interfaz inicial donde el usuario puede seleccionar que aplicación usar. Permite elegir entre diferentes datasets predeterminados (como CIFAR-10, MNIST, Fashion-MNIST) y el número de imágenes a clasificar. También permite la carga de imágenes propias para la clasificación. En la Figura 2.4 se muestra la interfaz web.

Listing 2.14: Ejemplo de input de etiquetas e imágenes en html

```
<form id="customLabelForm" action="/classify_custom"
method="post" enctype="multipart/form-data">
  <label for="custom_labels_only">
    Write Your Own Labels (comma-separated):</label>
  <input type="text" id="custom_labels_only" name="custom_labels_only"
    placeholder="e.g., _cat,_dog,_car" required>
  <label for="file_custom">Upload an Image:</label>
  <input type="file" id="file_custom" name="file_custom">
```

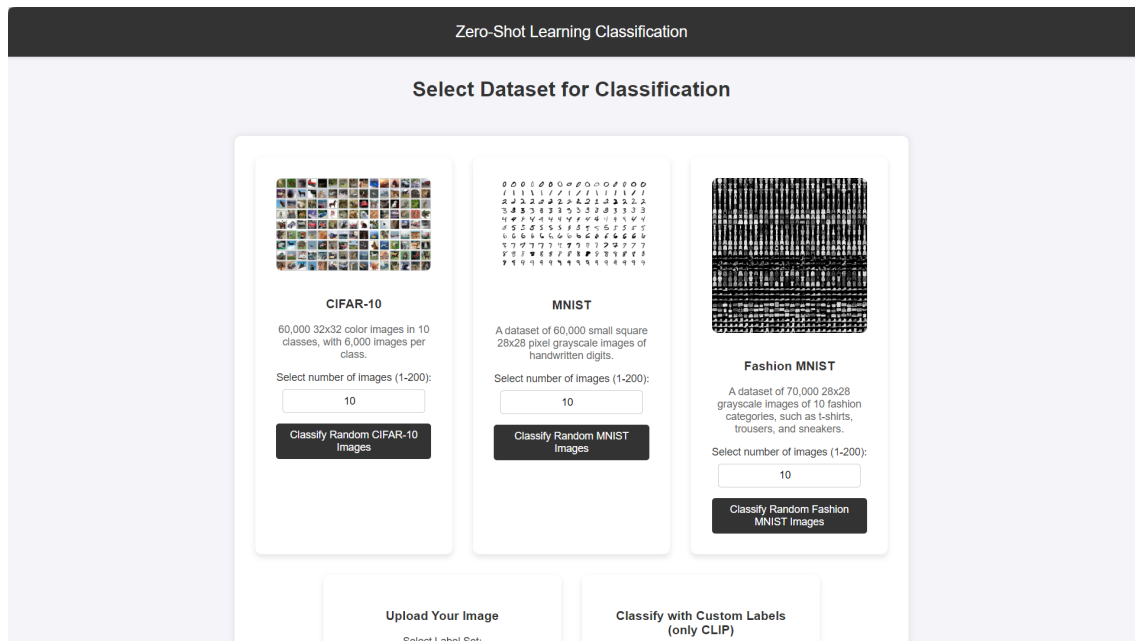


Figura 2.4: Interfaz de Usuario

```

    accept="image/*" required>
    <button type="submit">Classify Image with Custom Labels</button>
</form>

```

upload_result.html Esta plantilla muestra los resultados de la clasificación cuando el usuario carga una imagen propia. Los resultados incluyen las predicciones de los modelos ResNet50, VGG16 y CLIP, junto con los tiempos de clasificación y la imagen cargada.

Listing 2.15: Ejemplo de output de imagen y resultados en html

```

<h2>Classified Image</h2>

<h2>Predictions</h2>
<ul>
    {% for label, prob in result.clip_predictions %}
    <li class="prediction">{{ label }}
        <span>{{ "%.2f" % (prob * 100) }}%</span></li>
    {% endfor %}
</ul>
<h2>Processing Time</h2>
<p>{{ "%.2f" % result.clip_time }} seconds</p>
<a href="/" class="back-button">Go back</a>

```

classification_results.html Esta plantilla presenta los resultados de la clasificación de un dataset predeterminado. Muestra las imágenes clasificadas, las

Capítulo 2. Desarrollo

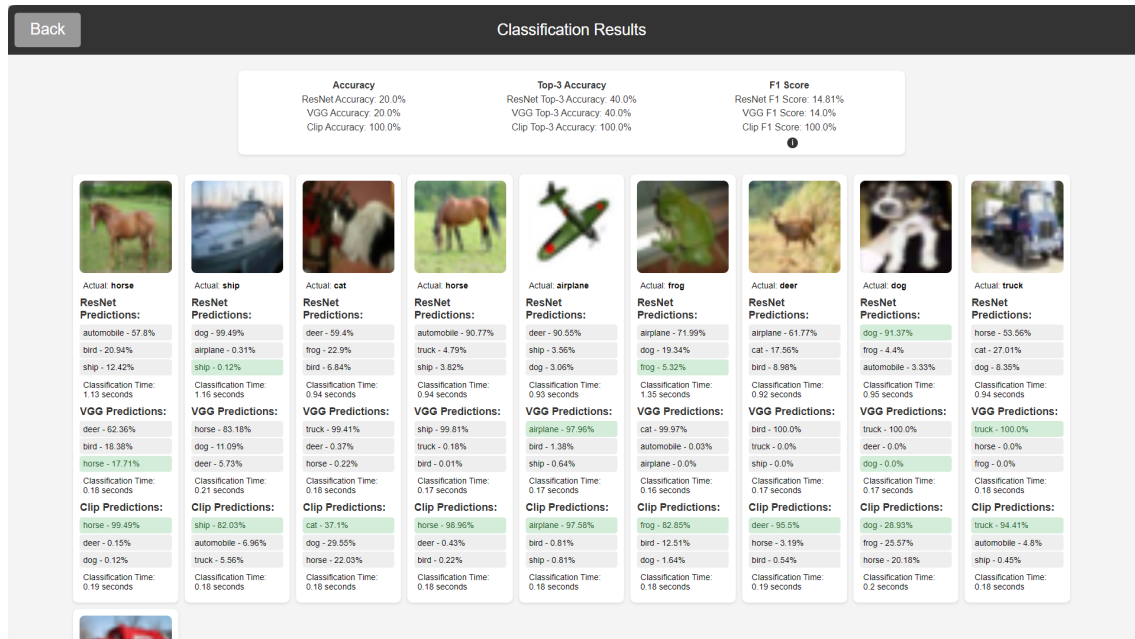


Figura 2.5: Resultados de la clasificación. En verde se muestra la opción correcta. El botón de volver dirige de nuevo a la selección de datasets.

predicciones de los modelos, la exactitud, la exactitud en el top-3, y la puntuación F1 para cada modelo utilizado (ResNet50, VGG16 y CLIP).

classify_custom.html Esta plantilla se utiliza para mostrar los resultados de la clasificación con etiquetas personalizadas. El usuario carga una imagen y define sus propias etiquetas, y la plantilla muestra las predicciones del modelo CLIP basado en esas etiquetas personalizadas.

La interfaz web permite a los usuarios seleccionar datasets predeterminados para la clasificación de imágenes y cargar imágenes propias para la clasificación utilizando etiquetas personalizadas. A continuación se describe el diseño y la funcionalidad de la interfaz.

2.10.2. Diseño de la Página

La página está diseñada con un contenedor central que contiene tarjetas (cards) para cada una de las funcionalidades disponibles. Cada tarjeta incluye un título, un formulario de entrada y un botón para enviar los datos.

2.10.3. Técnicas de Diseño para la Usabilidad

La página `classification_results.html` para mostrar los resultados de la clasificación de imágenes se ha diseñado teniendo en cuenta varias técnicas orientadas a mejorar la usabilidad. Estas técnicas aseguran que los usuarios puedan interactuar eficazmente con la interfaz, comprendiendo rápidamente la información presentada.

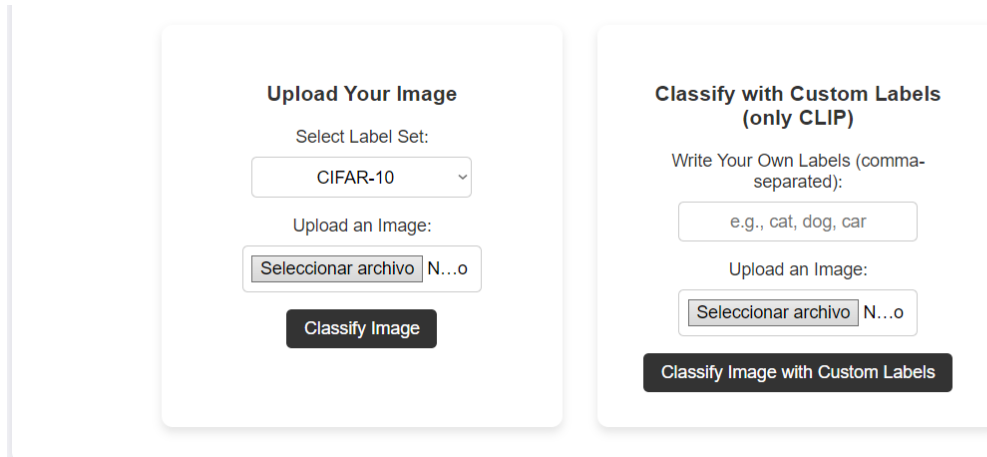


Figura 2.6: Selección de conjuntos de datos: Acceso a las aplicaciones de Clasificación con Imágenes Propias y Clasificación con Etiquetas Propias

Diseño Centrado en el Usuario

El diseño de la página se centra en la experiencia del usuario mediante el uso de una disposición clara y concisa, fácil de navegar y entender. A continuación se describen las técnicas específicas empleadas:

Uso de Encabezados y Botones Intuitivos

```
<header>
  <a href="" class="back-button">Back</a>
  Classification Results
</header>
```

Todas las páginas de la aplicación incluyen un botón para volver a la selección de datasets, que hace las veces de menú o pantalla principal. En este ejemplo, el encabezado incluye un botón de "Back"(Regresar) claramente visible, facilitando la navegación y mejorando la accesibilidad.

Diseño Responsivo para Usabilidad en Distintos Dispositivos

```
body {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

El diseño es responsivo, utilizando `flexbox` para centrar el contenido y asegurar que se adapte bien a diferentes tamaños de pantalla, lo que permite la accesibilidad a la aplicación tanto en dispositivos móviles como de escritorio.

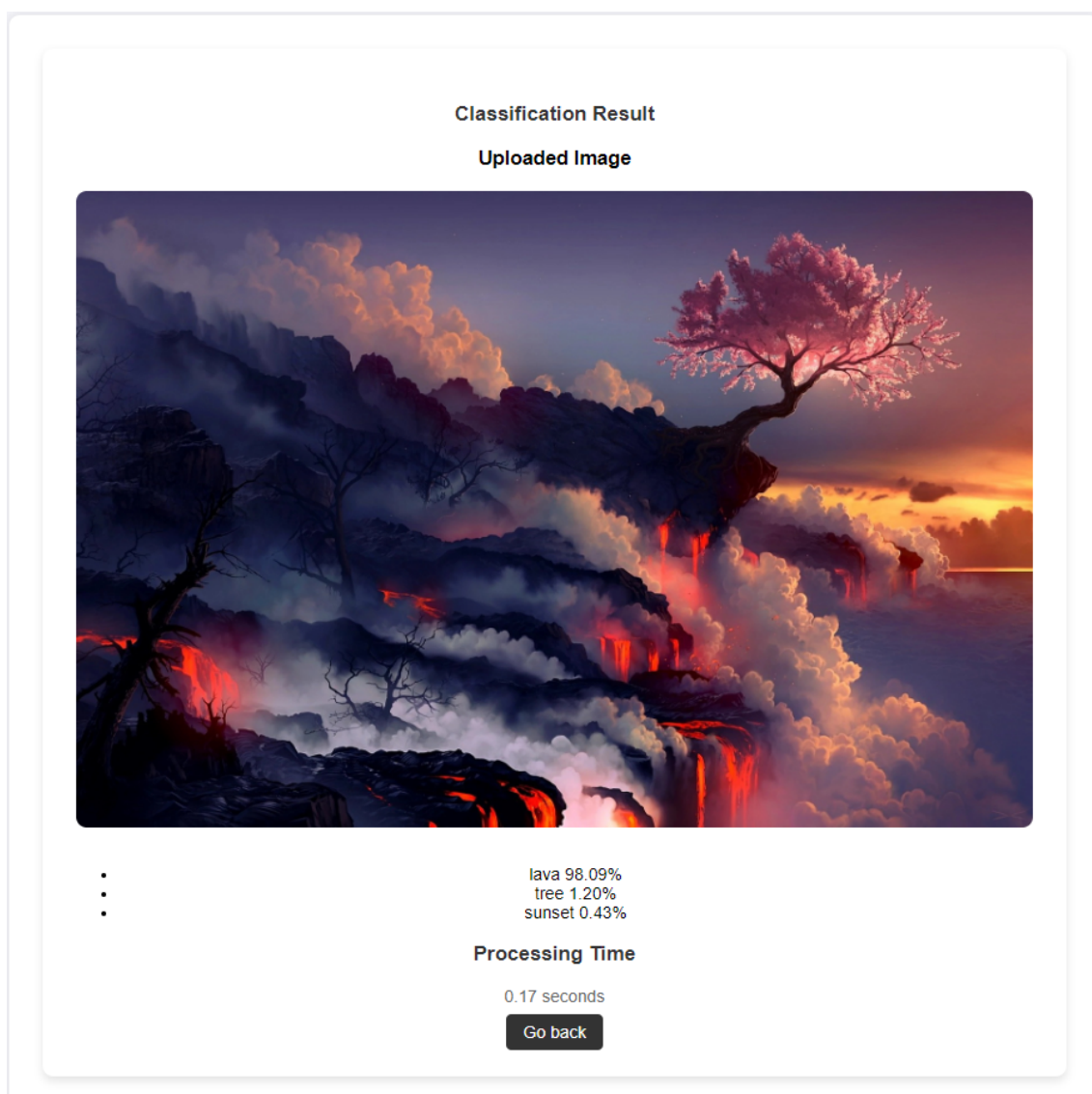


Figura 2.7: Resultados de Clasificación con Etiquetas Propias

Información Resumida en un Formato Claro

La sección de resumen presenta la precisión, precisión top-3 y el puntaje F1 de los modelos en un formato limpio y fácilmente comprensible, utilizando columnas para organizar la información.

Icono de Información y Tooltips

```
.info-icon {  
    display: inline-block;  
    margin-left: 10px;  
    width: 16px;  
    height: 16px;  
    background-color: #333;  
    color: #fff;  
    border-radius: 50%;  
    text-align: center;  
    line-height: 16px;  
    font-size: 12px;  
    cursor: pointer;  
    position: relative;  
}  
.info-icon:hover .tooltip {  
    display: block;  
}
```

Los iconos de información con tooltips proporcionan explicaciones adicionales sin sobrecargar visualmente la interfaz, mejorando la comprensión del usuario sobre las métricas presentadas.

Tarjetas de Imagen con Información Detallada

Las tarjetas de imagen muestran las imágenes clasificadas junto con las predicciones de cada modelo, con un diseño limpio y consistente que facilita la comparación visual. Las imágenes se presentan con un tamaño adecuado y bordes redondeados para una mejor estética. 2.5

Diferenciación Visual de Predicciones Correctas e Incorrectas

```
<p  
class="{[_'correct' _if_pred[0] _==_labels.index(result.actual_label)  
else_'incorrect'_]}">
```

Las predicciones correctas e incorrectas se destacan visualmente mediante el uso de colores diferentes, lo que permite a los usuarios identificar rápidamente los resultados correctos.

Adaptación del Diseño al Número de Imágenes Clasificadas

```
.container {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));  
  gap: 10px;  
  width: 100%;  
  padding: 10px;  
  padding-left: 100px;  
  padding-right: 100px;  
  box-sizing: border-box;  
}
```

El uso de una disposición en grid asegura que las tarjetas de imagen se organicen de manera fluida y ordenada, adaptándose al tamaño del contenido y del contenedor, mejorando así la legibilidad y la navegación.

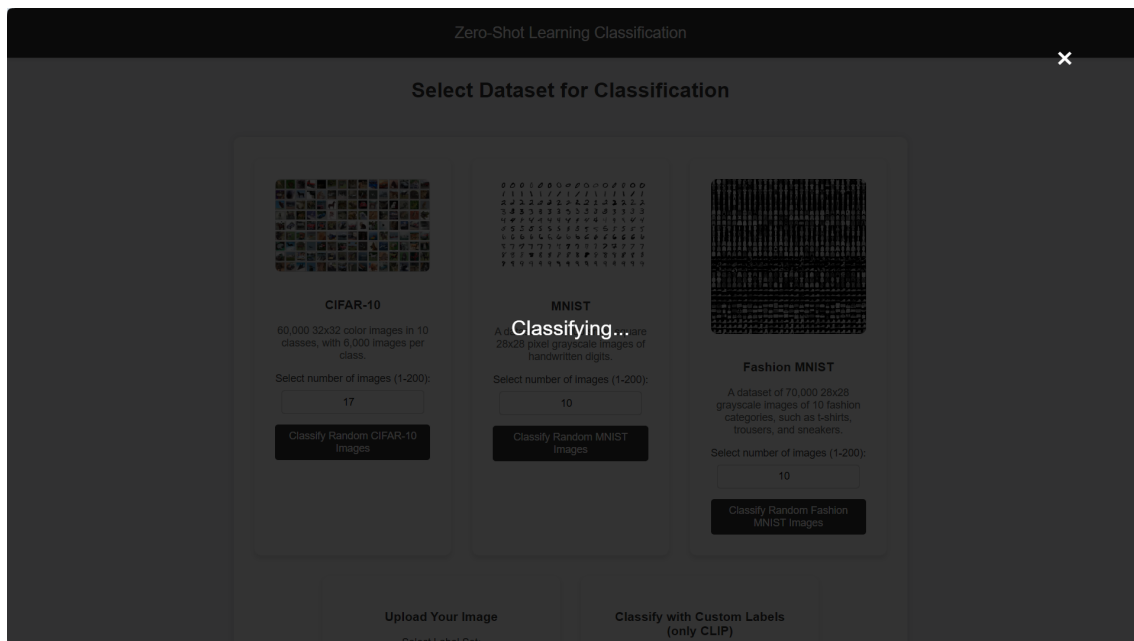


Figura 2.8: Cubrepantallas mientras la aplicación realiza el proceso de clasificación. Se incluye un botón "x" para poder detener la clasificación en cualquier momento.

Capítulo 3

Análisis de impacto

3.0.1. Impacto del Proyecto

Personal

La realización de este trabajo de fin de grado tiene un significativo impacto personal en el estudiante. Entre los beneficios esperados se incluyen:

- **Desarrollo de Habilidades Técnicas:** El estudiante adquiere competencias avanzadas en programación, aprendizaje automático, procesamiento de imágenes y desarrollo web.
- **Experiencia Práctica:** La aplicación práctica de conocimientos teóricos en un proyecto real mejora la capacidad del estudiante para enfrentar desafíos técnicos y resolver problemas.
- **Portafolio Profesional:** El proyecto puede ser añadido al portafolio del estudiante, mejorando sus perspectivas de empleo y demostrando su capacidad para llevar a cabo proyectos complejos.

Empresarial

Desde una perspectiva empresarial, el proyecto presenta varios beneficios potenciales:

- **Aplicabilidad en Diversos Sectores:** Las técnicas de clasificación de imágenes pueden ser útiles en sectores como la salud, la seguridad, el comercio y la agricultura.
- **Mejora de Procesos:** La automatización de tareas mediante la clasificación de imágenes puede mejorar la eficiencia operativa y reducir costos.

Social

El impacto social de este proyecto puede ser considerable:

- **Acceso a Tecnología Avanzada:** Facilitar el acceso a herramientas de clasificación de imágenes puede democratizar el uso de tecnologías avanzadas.

Capítulo 3. Análisis de impacto

- **Educación y Formación:** El proyecto puede servir como recurso educativo para otros estudiantes y profesionales interesados en el aprendizaje automático y la inteligencia artificial.

Económico

El impacto económico del proyecto no es tan significativo, aunque con algunas modificaciones podría serlo, por las siguientes razones:

- **Reducción de Costos:** La automatización de tareas de clasificación puede reducir los costos laborales y aumentar la eficiencia.
- **Aumento de la Productividad:** La implementación de soluciones automatizadas puede aumentar la productividad en diversos sectores industriales.

Medioambiental

Aunque no es el foco principal del proyecto, puede tener implicaciones medioambientales:

- **Eficiencia Energética:** El uso de modelos eficientes y optimizados puede reducir el consumo de energía en comparación con métodos manuales o menos eficientes.
- **Aplicaciones Ecológicas:** La clasificación de imágenes puede ser utilizada en aplicaciones medioambientales, como el monitoreo de fauna y flora, contribuyendo a la conservación ambiental.

Cultural

El impacto cultural del proyecto puede incluir:

- **Promoción de la Cultura Científica:** Proyectos de este tipo promueven la cultura científica y tecnológica en la sociedad.
- **Intercambio de Conocimientos:** La difusión de los resultados del proyecto puede fomentar el intercambio de conocimientos y la colaboración entre diferentes disciplinas y culturas.

Relación con los Objetivos de Desarrollo Sostenible (ODS)

El proyecto puede contribuir a varios ODS de la Agenda 2030:

- **ODS 4 - Educación de Calidad:** Al proporcionar una herramienta educativa y de aprendizaje, el proyecto contribuye a una educación inclusiva y de calidad.
- **ODS 8 - Trabajo Decente y Crecimiento Económico:** La automatización y mejora de procesos pueden fomentar el crecimiento económico y la productividad.

-
- **ODS 9 - Industria, Innovación e Infraestructura:** El desarrollo de tecnologías avanzadas impulsa la innovación y la construcción de infraestructuras resilientes.
 - **ODS 12 - Producción y Consumo Responsables:** El uso eficiente de recursos tecnológicos contribuye a patrones de producción y consumo sostenibles.

El **zero-shot learning** permite la clasificación de imágenes sin necesidad de recopilar y etiquetar grandes conjuntos de datos para cada nueva tarea. Esto reduce significativamente el consumo de recursos, tanto en términos de tiempo como de energía. Además, se ha seleccionado y configurado modelos de clasificación de imágenes como ResNet50 y VGG16, que son conocidos por su eficiencia en términos de precisión y velocidad. Estos modelos han sido preentrenados y optimizados para reducir el consumo de recursos computacionales durante la clasificación.

Estas decisiones estratégicas demuestran un compromiso con el ODS 12, enfocándose en la producción y el consumo responsables mediante el uso de tecnologías avanzadas que optimizan la eficiencia y minimizan el impacto ambiental. La adopción de zero-shot learning y enfoques sin entrenamiento representa un paso significativo hacia una inteligencia artificial más sostenible y consciente del medio ambiente.

- **ODS 13 - Acción por el Clima:** Las aplicaciones ecológicas de la clasificación de imágenes pueden ayudar en la acción contra el cambio climático.

Capítulo 4

Conclusiones y trabajo futuro

4.1. Conclusiones

Este código proporciona una plataforma versátil para la clasificación de imágenes, utilizando tanto datasets y etiquetas predeterminadas como imágenes y etiquetas personalizadas. La combinación de modelos preentrenados y una interfaz web sencilla hace que sea una herramienta flexible para aplicaciones de clasificación de imágenes.

El objetivo de este proyecto era diseñar y desarrollar un sistema automatizado que pudiera identificar el modelo preentrenado más adecuado para una tarea específica y un conjunto de datos determinado. A pesar de los esfuerzos realizados y los avances alcanzados en varias áreas, no se logró cumplir completamente este objetivo.

No obstante, el resultado final permite hacer una decisión informada sobre la elección del modelo. Logrando una versión no automática, sino manual, del objetivo propuesto.

Desarrollar un sistema completamente automatizado no es trivial a partir de los resultados de este trabajo.

La elección de la mejor métrica para comparar dos modelos de clasificación de imágenes depende del contexto específico y de los objetivos del proyecto. Si las clases están equilibradas, la precisión (accuracy) puede ser suficiente. Si hay un desbalance de clases, el F1 Score puede ser más representativo. En problemas donde las probabilidades son importantes, otras métricas como Log-Loss y AUC-ROC pueden ser más adecuadas.

4.1.1. Complejidad Técnica

Uno de los principales desafíos enfrentados fue la complejidad técnica inherente a la identificación automática del modelo más adecuado. Este proceso implica no solo una comprensión profunda de las características de los modelos preentrenados, con la que no se contaba al empezar al proyecto, y que se ha tenido

Capítulo 4. Conclusiones y trabajo futuro

que construir progresivamente desde cero, sino también la capacidad de analizar y comparar una amplia variedad de tareas y conjuntos de datos. La falta de experiencia en el campo hizo que el desarrollo de un sistema totalmente automatizado y preciso fuera más complicado de lo anticipado.

4.1.2. Limitaciones de Tiempo

Durante el transcurso del proyecto se ha puesto en evidencia que el desarrollo de un sistema de esta envergadura requiere una cantidad significativa de tiempo y recursos. A pesar de la planificación y el esfuerzo dedicado, no se han alcanzado los aspectos necesarios para alcanzar el objetivo inicial.

4.1.3. Desafíos en la Integración de Datos

Otro obstáculo importante fue la integración de diferentes tipos de datos y la necesidad de estandarizar estos datos para su análisis. Por ello se optó por utilizar datasets preparados, para disponer rápidamente de un número razonable de imágenes etiquetadas para realizar pruebas. Esta decisión probó ser eficaz para las pruebas, pero provocó que las clasificaciones realizadas fuesen menos interesantes, por ello se optó por desarrollar una aplicación complementaria que permitiese clasificar cualquier imagen.

4.1.4. Resultados de la Comparación de modelos

Los modelos elegidos para hacer la comparación se cuentan entre los más populares, sin embargo, la comparación hecha no permite sacar conclusiones útiles para el estado del arte, ya que las técnicas utilizadas son distintas, y en concreto, como podemos observar en este trabajo, el aprendizaje por transferencia no es un buen acercamiento a problemas de Zero-Shot Learning.

Sin embargo, la implementación web permite realizar comparaciones de manera eficiente y en diferentes contextos, facilitando la experimentación con diversas arquitecturas y enfoques de aprendizaje. por ejemplo, se podría utilizar esta herramienta para comparar un Google-ViT adaptado para Zero-Shot con CLIP, lo que permitiría evaluar cómo un modelo específicamente diseñado para imágenes se comporta en comparación con un modelo multimodal en tareas de zero-shot learning.

Al final, la plataforma web puede ayudar a los usuarios a entender mejor los resultados y a tomar decisiones informadas sobre la selección de modelos.

4.2. Trabajo Futuro

4.2.1. Intercambio Sencillo de Modelos, con posible integración de una API como Huggingface

Implementar la capacidad de importar y utilizar modelos directamente desde la API de Huggingface, facilitando el intercambio y actualización de modelos preentrenados. El objetivo sería proporcionar una interfaz amigable para seleccionar

y cargar modelos desde Huggingface, permitiendo a los usuarios experimentar con diferentes arquitecturas y versiones de modelos.

4.2.2. Más Métricas

Incluir métricas adicionales como el log-loss, la curva ROC y el AUC (Área Bajo la Curva) para proporcionar una evaluación más completa del rendimiento del modelo. Para ello también sería necesario desarrollar herramientas de visualización para representar gráficamente estas métricas y facilitar la interpretación.

4.2.3. Cómputo de Memoria Utilizada

Implementar funciones para monitorear y registrar el uso de memoria durante el proceso de clasificación, proporcionando datos sobre la eficiencia de los modelos y poder así realizar decisiones informadas. Utilizar estos datos para optimizar los modelos y reducir el consumo de recursos, mejorando la sostenibilidad del sistema.

4.2.4. Posibilidad de Subir Varias Imágenes o Datasets desde la Web

Desarrollar la funcionalidad para permitir a los usuarios cargar múltiples imágenes simultáneamente, facilitando la evaluación de modelos en lotes de datos.

Otra idea es la de permitir la carga y gestión de múltiples datasets directamente desde la interfaz web, incluyendo opciones para etiquetar y organizar los datos. Así como integrar herramientas para preprocesar y explorar los datasets cargados, mejorando la facilidad de uso y la flexibilidad del sistema.

4.2.5. Soporte Multilingüe

: Implementar soporte multilingüe para hacer la aplicación accesible a una audiencia global.

4.3. Comentario Personal

Personalmente, estoy satisfecho con el trabajo realizado, pero no tanto con los resultados del mismo. Es muy probable que siga desarrollando esta aplicación, intentando implementar alguna de las mejoras propuestas.

Este trabajo no buscaba crear o difundir técnicas ni métodos innovadores. Su objetivo es demostrar que se han adquirido las competencias y resultados de aprendizaje correspondientes a la asignatura "Trabajo Fin de Grado" del "Grado en Matemáticas e Informática".

Este objetivo en mi opinión ha sido cumplido, ya que se han aplicado adecuadamente técnicas de desarrollo aprendidas durante la carrera, como la metodología ágil, el desarrollo de software web, y la implementación de inteligencia artificial.

Bibliografía

- [1] “Learn Python, Data Viz, Pandas more | Tutorials | Kaggle.” [Online]. Available: <https://www.kaggle.com/learn>
- [2] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensor-Flow, 2nd Edition*. O'Reilly Media, Inc., 2019.
- [3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *neural information processing systems*, vol. 25, pp. 1097–1105, 12 2012. [Online]. Available: http://books.nips.cc/papers/files/nips25/NIPS2012_0534.pdf
- [4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [5] L. C. James Briggs, “Embedding Methods for Image Search.” [Online]. Available: <https://www.pinecone.io/learn/series/image-search/>
- [6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1 1998. [Online]. Available: <https://doi.org/10.1109/5.726791>
- [7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [9] X. Li, G. Zhang, H. H. Huang, Z. Wang, and W. Zheng, “Performance analysis of gpu-based convolutional neural networks,” in *2016 45th International Conference on Parallel Processing (ICPP)*, 2016, pp. 67–76.
- [10] K. Team, “Keras documentation: Keras Applications.” [Online]. Available: <https://keras.io/api/applications/>
- [11] “¿En qué consiste el aumento de datos?: Explicación sobre las técnicas de aumento de datos: AWS.” [Online]. Available: <https://aws.amazon.com/es/what-is/data-augmentation/#:>

- ~:~text=Data%20augmentation%20is%20the%20process,machine%20learning%20(ML)%20models.
- [12] C. Dilmegani, "Top Data Augmentation Techniques: Ultimate Guide for 2024," 2 2024. [Online]. Available: <https://research.aimultiple.com/data-augmentation-techniques/>
- [13] S. Singh, "What is data augmentation? Techniques, examples amp; benefits," 11 2022. [Online]. Available: <https://www.labellerr.com/blog/what-is-data-augmentation-techniques-examples-benefits/>
- [14] "Pre Trained Model Definition | ENCORD." [Online]. Available: <https://encord.com/glossary/pre-trained-model-definition/#:~:text=Pre-trained%20models%20are%20often%20trained%20on%20large%2C%20diverse%20datasets,the%20performance%20of%20the%20model.>
- [15] A. Singh, "Leveraging Zero-Shot, Few-Shot, and Transfer Learning: A Comprehensive Guide for Enterprise AI," 10 2023. [Online]. Available: <https://medium.com/@anand94523/leveraging-zero-shot-few-shot-and-transfer-learning-a-comprehensive-guide-for-enterpris>
- [16] A. H. Syed, "Transfer learning and few-shot learning: Improving generalization across diverse tasks and domains," 4 2023. [Online]. Available: <https://syedabis98.medium.com/transfer-learning-and-few-shot-learning-improving-generalization-across-diverse-tasks-a>
- [17] A. Bosco, "Transfer Learning: Feature Extraction and Fine tuning," 3 2024. [Online]. Available: <https://medium.com/data-reply-it-datatech/transfer-learning-feature-extraction-and-fine-tuning-db7d82767992#:~:text=Fine-tuning%20is%20more%20flexible,when%20labeled%20data%20is%20limited.>
- [18] "Transfer learning and fine-tuning." [Online]. Available: https://www.tensorflow.org/tutorials/images/transfer_learning
- [19] F. Hvilshøj, "What is One-Shot Learning in Computer Vision," 3 2023. [Online]. Available: <https://encord.com/blog/one-shot-learning-guide/#:~:~text=Unlike%20traditional%20computer%20vision%20projects,data%20to%20compare%20images%20against.>
- [20] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.
- [21] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/cifar.html>
- [22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998. [Online]. Available: <http://yann.lecun.com/exdb/mnist>

- [23] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017. [Online]. Available: <https://arxiv.org/abs/1708.07747>

Anexos

Apéndice A

Primer anexo: Índice de originalidad

Capítulo A. Primer anexo: Índice de originalidad

tfg_latex_etsiinf_2023_02_20 (1).pdf

INFORME DE ORIGINALIDAD

18%

INDICE DE SIMILITUD

15%

FUENTES DE INTERNET

6%

PUBLICACIONES

11%

TRABAJOS DEL
ESTUDIANTE

FUENTES PRIMARIAS

1

Submitted to Universidad Politécnica de Madrid

Trabajo del estudiante

2%

2

aws.amazon.com

Fuente de Internet

1%

3

oa.upm.es

Fuente de Internet

1%

4

arxiv.org

Fuente de Internet

1%

5

Submitted to Universidad Internacional de la Rioja

Trabajo del estudiante

1%

6

Submitted to Texas A & M University, Kingville

Trabajo del estudiante

<1%

7

Submitted to Universidad Anahuac México Sur

Trabajo del estudiante

<1%

8

ikee.lib.auth.gr

Fuente de Internet

<1%