UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Generación de explicaciones basadas en ejemplos de modelos de predicción de objetos en garabatos

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

AUTOR/A: Cheng Liu, Juan Jianxin

Tutor/a: Ferri Ramírez, César

Cotutor/a: Hernández Orallo, José

CURSO ACADÉMICO: 2022/2023

Resumen:

Este proyecto se centra en la exploración del campo de la Explicabilidad de la Inteligencia Artificial (XAI), que busca abordar los desafíos asociados con la comprensión de las decisiones tomadas por los sistemas de IA.

En este contexto, el TFM el enfoque principal es demostrar que la simplicidad en los dibujos facilita el proceso de aprendizaje, lo que contribuye a su transparencia y comprensión. Se explorarán modelos de aprendizaje diseñados específicamente para abordar el desafío de la detección de objetos a partir de trazos de dibujos realizados por usuarios. Estos modelos aplicarán técnicas de aprendizaje para identificar patrones y características en los trazos de objetos, comparándolos con patrones conocidos en una base de datos para determinar la naturaleza de los objetos dibujados.

Resum:

Aquest projecte se centra en l'exploració del camp de l'Explicabilidad de la Intel·ligència Artificial (XAI), que busca abordar els desafiaments associats amb la comprensió de les decisions preses pels sistemes de IA.

En aquest context, el TFM l'enfocament principal és demostrar que la simplicitat en els dibuixos facilita el procés d'aprenentatge, la qual cosa contribueix a la seua transparència i comprensió. S'exploraran models d'aprenentatge dissenyats específicament per a abordar el desafiament de la detecció d'objectes a partir de traços de dibuixos realitzats per usuaris. Aquests models aplicaran tècniques d'aprenentatge per a identificar patrons i característiques en els traços d'objectes, comparant-los amb patrons coneguts en una base de dades per a determinar la naturalesa dels objectes dibuixats.

Abstract:

This project focuses on exploring the field of Explainable Artificial Intelligence (XAI), which aims to address the challenges associated with understanding the decisions made by AI systems.

In this context, the main focus of the Master's Thesis (TFM) is to demonstrate that simplicity in drawings facilitates the learning process, contributing to transparency and understanding. Specific learning models designed to tackle the challenge of object detection from user-drawn sketches will be explored. These models will apply learning techniques to identify patterns and features in the strokes of objects, comparing them with known patterns in a database to determine the nature of the drawn objects.

Índice General

Tabla de imágenes

Tabla de gráficas

1. Introducción

1.1 Contexto y motivación

Quick, Draw!, un proyecto de Google que involucra inteligencia artificial es un juego online que trata de dibujar-adivinar mediante la interacción del usuario, el cual es el encargado de dibujar la categoría que indica el juego en un tiempo limitado, y conforme se va completando el dibujo, el modelo de aprendizaje automático intentará adivinar el objeto dibujado. En caso de acertar el dibujo, directamente se pasará a la siguiente categoría a dibujar aunque puede darse el caso de que el modelo juego no consiga adivinar el dibujo, en cuyo caso se pasará de categoría transcurrido el tiempo. Independientemente de si el juego adivina o no la categoría dibujada, este se almacenará en la base de datos del juego para que el modelo de red neuronal aprenda del dibujo.

Todos los dibujos realizados por los jugadores se almacenan en una base de datos, que actualmente llega a los 50 millones de dibujos recopilados de unas 345 categorías que ofrece el juego.[4] Esta base de datos se ha convertido en un recurso valioso para la investigación en reconocimiento de patrones y del aprendizaje automático. Dado que los modelos de aprendizaje se han vuelto cada vez más complejos y precisos, y por lo tanto también se vuelven más complicados de investigar e interpretar.

La explicabilidad de los modelos se ha vuelto un tema crucial en todo lo relacionado con el aprendizaje automático debido a la necesidad de comprender cómo y por qué motivo toman ciertas decisiones los modelos. Entonces la generación de explicaciones es una de las soluciones que puede abordar la falta de transparencia en los modelos de aprendizaje automático.

Este es uno de los motivos por el cual se ha decidido abordar dicho tema, el hecho de que con el creciente uso de la inteligencia artificial, la explicabilidad puede llegar a mejorar la confianza de los usuarios gracias a que transmite comprensión, transparencia y confianza. Por lo tanto, es de suma importancia que los ejemplos utilizados siempre tiendan a ser lo más simples y claros posibles, dado que en caso de que se utilicen ejemplos demasiado complejos o técnicos, puede resultar complicado que el usuario logre comprender la toma de

decisiones de la inteligencia artificial o incluso que no se logre el objetivo de la explicabilidad.

1.2 Objetivos

El objetivo de este trabajo consistirá en realizar explicaciones de los sistemas de inteligencia artificial basados en ejemplos, tratando de demostrar que mediante los ejemplos más sencillos se logra un mayor aprendizaje. Para ello, se realizará un entrenamiento de dos modelos, uno será basado en Redes neuronales convolucionales (CNN) y otro en Random Forest (RF). La base de entrenamiento de ambos modelos será la base de datos de Google, donde únicamente se seleccionarán los datos de 5 parejas de categorías que resulten similares. A partir de dichos datos se realizarán los entrenamientos, el primer modelo se entrenará con todo el conjunto de datos de la pareja de categorías, y una vez entrenado, se utilizará para realizar un primer filtrado de aquellos dibujos que logre identificar correctamente el modelo. Esto se realiza para evitar utilizar dibujos que sean irreconocibles, puesto que hay dibujos que pueden pertenecer a una categoría y no tengan nada que ver con ella. Y posteriormente se realiza un ordenado en función del grado de simplicidad de las imágenes, esto realizado a partir del número de trazos realizados por los usuarios.

Una vez filtradas las imágenes, estas se recogen y se utilizarán para entrenar el segundo modelo, para el cual se utilizan un número limitado de imágenes para su entrenamiento, escogiendo siempre los primeros elementos, dado que gracias al ordenamiento anterior, los primeros siempre serán aquellos que sean más simples. El número de imágenes utilizadas irán variando de 10 a 30.000, con el fin de comprobar la evolución del porcentaje de acierto del modelo al reconocer las categorías de las imágenes en función del número de imágenes utilizadas en su entrenamiento.

A parte, se recogerán un número de imágenes, teniendo en cuenta diversos factores, como la simplicidad o la aleatoriedad, para realizar pruebas con sujetos para ver qué tan efectivos son los humanos identificando imágenes en comparación a los modelos. Y posteriormente con dichos resultados se realizará un análisis para poder comprender la explicabilidad que aportan los bocetos más simples.

1.3 Estructura del TFM

En este trabajo, la estructura está separada de tal forma, con el fin de dar una compresión detallada de la investigación llevada a cabo, basado en el ámbito de la generación de explicaciones basadas en ejemplos de modelos de predicción de objetos basados trazos o bocetos. A continuación, se indica la organización de cada sección:

1. Introducción

En esta sección, se habla sobre el contexto de la investigación, presentando la relevancia de la generación de explicaciones en el ámbito de los modelos de predicción de objetos en garabatos. A parte, también se plantean los objetivos y la motivación detrás de la investigación destacando la importancia de comprender y comunicar el proceso de toma de decisiones de los modelos de aprendizaje automático.

2. Marco Teórico

En el marco teórico, se exploran los conceptos fundamentales relacionados con la generación de explicaciones en modelos de predicción. Se abordan las técnicas como las redes neuronales convolucionales (CNN) y los algoritmos de los bosques aleatorios (RF). Y por último, se aborda el concepto de explicabilidad en los modelos de aprendizaje automático y su influencia en la confianza de los usuarios para su uso.

3. Estado del Arte

En esta sección, se realiza una revisión de trabajos previos relacionados con la generación de explicaciones en modelos de predicción de garabatos. Analizando diferentes enfoques, métodos y herramientas utilizados en los modelos y en la generación de explicaciones.

4. Conjunto de Datos

En el capítulo del conjunto de datos se proporciona una descripción detallada de la base de datos utilizada en la investigación, QuickDraw. Se aborda cómo se recopilaron los garabatos, las parejas que han sido seleccionadas para ser

utilizadas en el entrenamiento de los modelos y como finalmente se ha preparado y limpiado el conjunto de datos previo al entrenamiento de los modelos.

5. Metodología

En este capítulo, se explica en profundidad la arquitectura de los modelos utilizados, tanto en la red neuronal convolucional (CNN) como el Random Forest (RF). Se describe el diseño experimental detallando el proceso de selección de las características y variables controladas.

6. Resultados

La sección de resultados se presentan los métodos de generación de resultados utilizados en todas las muestras. Se muestran y analizan los resultados obtenidos del modelo CNN y en los resultados del modelo RF construido a partir de las explicaciones generadas por la CNN. Se muestran también los resultados obtenidos a partir de las pruebas de sujetos reales.

7. Discusión de resultados

En la discusión de resultados se realiza una síntesis e interpretación de los resultados obtenidos en las pruebas de los modelos y mediante sujetos humanos. Se discute cómo la generación de explicaciones puede influir en la interpretación y confianza en los modelos de predicción de garabatos. Y finalmente se realiza un análisis de las limitaciones del proyecto.

8. Conclusiones

En las conclusiones, se realiza un resumen respecto a los resultados evaluando los objetivos iniciales de la investigación. Se reflexiona sobre las contribuciones y posibles direcciones futuras de la investigación.

9. Anexo

En este aparatado se van a mostrar los detalles de la implementación llevada a cabo en el proyecto, donde se mostrará mediante pseudocódigo las partes imprescindibles. A parte se detallarán los objetivos de Desarrollo Sostenible empleados en este proyecto.

10. Bibliografía

El capítulo de bibliografía recopila todas las referencias utilizadas a lo largo del TFM, brindando una base sólida para la investigación.

2. Marco Teórico

En este capítulo, se proporciona una base teórica sobre los conceptos fundamentales del Aprendizaje Automático y las Redes Neuronales, centrándose en el Aprendizaje Supervisado, las Redes Neuronales Convolucionales (CNN), los Modelos de Random Forest (RF) y la importancia de la Explicabilidad en los modelos de Inteligencia Artificial.

2.1 Aprendizaje Automático

El aprendizaje automático es un subconjunto de inteligencia artificial que permite que un sistema aprenda y mejore de forma autónoma mediante redes neuronales y aprendizaje profundo, sin tener que ser programado explícitamente, a través de la ingesta de grandes cantidades de datos.[11] El uso del aprendizaje automático supone un avance significativo en cuanto a la capacidad que poseen las máquinas de adquirir conocimiento, ya que permite a las máquinas aprender de forma autónoma patrones y relaciones a partir de los datos proporcionados, lo que facilita la toma de decisiones y la generación de predicciones.

Por otro lado, tenemos el modelo, parte principal de cualquier dispositivo de aprendizaje automático. Un modelo es un algoritmo cuyo objetivo es aprender patrones a partir de los datos disponibles para realizar tareas específicas, como clasificación, regresión, agrupamiento y más. Los modelos de aprendizaje pueden ser entrenados de dos formas, siendo una el Aprendizaje Supervisado en el que se entrenan utilizando datos etiquetados o el Aprendizaje No Supervisado en los que los datos utilizados no se encuentran etiquetados.

2.1.1 Aprendizaje Supervisado

El Aprendizaje Supervisado realiza un papel esencial en la creación de modelos capaces de realizar tareas de predicción. Este enfoque se basa en el principio de que un modelo puede aprender de a partir de una serie de ejemplos previamente etiquetados para realizar predicciones precisas a partir de otros datos nuevos no etiquetados.

En el Aprendizaje Supervisado, el proceso de entrenamiento implica proporcionar al modelo un conjunto de datos de entrenamiento compuesto por

una entrada y una salida conocida. Cada entrada representa una instancia de datos, y su salida correspondiente es la etiqueta. Entonces a partir del conjunto de datos o ejemplos etiquetados entregados al modelo, se va moldeando para que obtener finalmente el comportamiento deseado que se aplicará al ponerse a prueba el modelo con ejemplos nuevos sin etiquetar. Siendo el objetivo principal del Aprendizaje Supervisado aprender la relación entre las entradas y las salidas para que el modelo pueda generalizar y hacer predicciones precisas en datos no vistos previamente.

El Aprendizaje Supervisado se divide en dos categorías principales: Clasificación y Regresión.

- **Clasificación**: En la clasificación, el objetivo es predecir una etiqueta o categoría para cada entrada. Por ejemplo, en la clasificación de correos electrónicos como "spam" o "no spam", se espera que a partir de un correo se pueda identificar la etiqueta correspondiente.
- **Regresión**: En la regresión, el objetivo es predecir un valor en función de las características de entrada. Por ejemplo, predecir el precio de un objeto en función de sus características.

Pero en este trabajo, se va a enfocar principalmente en el uso del aprendizaje supervisado de Clasificación.

2.1.2 Redes Neuronales Convolucionales (CNN)

Inspirado por la estructura y el funcionamiento del cerebro humano emerge una clase de modelos, las Redes Neuronales, estas son una implementación concreta del Aprendizaje Automático y subcategoría de redes neuronales que hoy en día son uno de los modelos de clasificación de imágenes considerados como más eficaces.[16]

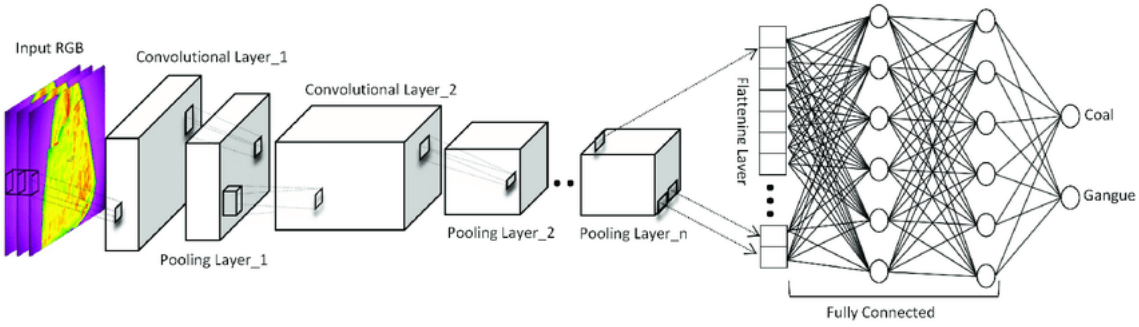


Figura 1.Capas de las Redes Neuronales Convolucionales [21]

Las Redes Neuronales se basan en una interconexión de nodos llamadas neuronas artificiales. Cada neurona realiza una operación simple en sus entradas y produce una salida, estando organizadas en capas, la capa de entrada recibe unos datos iniciales con los cuales van pasando entre capas, siendo todas ellas capas intermedias hasta llegar a la última capa, la capa de salida. En el paso de capa a capa se van realizando transformaciones de los datos hasta generar una salida final, dado lugar al resultado.

Estas redes están compuestas por dos partes fundamentales que trabajan en conjunto para procesar y clasificar imágenes:

- **Convolucional**: El objetivo principal de esta parte es extraer características distintivas de cada imagen y comprimirlas para reducir su tamaño inicial. La imagen de entrada pasa a través de una serie de filtros, cada uno de los cuales crea nuevas imágenes llamadas tarjetas de convoluciones. Finalmente, todos estos mapas de convolución se concatenan en un único vector de características llamado "código CNN".[16]
- **Clasificación**: El código CNN generado en la salida de la parte convolucional se introduce en una segunda parte de la red, que consta de capas totalmente conectadas conocidas como perceptrón multicapa (Multi Layers Perceptron). La función principal de esta parte es combinar las características extraídas por el código CNN para clasificar la imagen en una categoría específica.[16]

Las Redes Neuronales Convolucionales (CNN) son un tipo de red neuronal que ha generado una revolución en el campo del procesamiento de imágenes y la

visión por computadora. Estos modelos de redes neuronales están especialmente diseñados para trabajar con datos dispuestos en forma de cuadrícula, como imágenes y datos en 2D. Para lograrlo, se usan capas de convolución que les permiten detectar características como bordes y texturas e incorporan capas de agrupación para reducir la dimensionalidad de los datos. Gracias a todo esto, se ha podido realizar un avance e innovación en el campo del procesamiento visual, en tareas tal como la detección de objetos.

2.1.3 Modelos de Random Forest (RF)

Los Modelos de Random Forest (RF) representan un enfoque distintivo en el ámbito del aprendizaje automático, conocidos por su capacidad para realizar predicciones precisas y manejar datos complejos y ruidosos. Estos modelos son un tipo de combinación de árboles de decisión que logran obtener resultados más robustos y estables.

El concepto clave detrás de los RF radica en la combinación de múltiples árboles de decisión individuales. Aunque los árboles individuales pueden ser propensos al sobreajuste y a la inestabilidad, al combinar sus predicciones, los RF logran resultados más precisos y consistentes.[15]

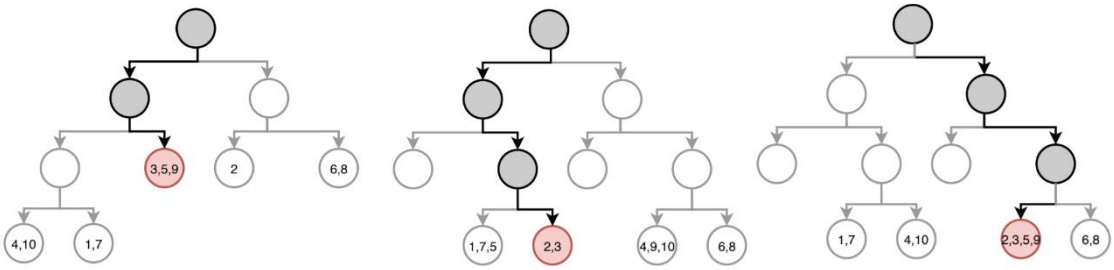


Figura 2. Árbol de Random Forest [12]

La construcción de un Random Forest implica dos fases fundamentales: la generación de muestras de entrenamiento mediante muestreo con reemplazo (bootstrapping) y la construcción de árboles de decisión para cada subconjunto de datos. Cada árbol emite una predicción y, en el caso de la clasificación, la clase más votada se convierte en la predicción final. En la regresión, se promedian las predicciones individuales de los árboles para obtener una predicción final.

2.2 Inteligencia Artificial Explicable

En este último apartado del marco teórico, abordamos un concepto de gran relevancia en el mundo de la inteligencia artificial: la Inteligencia Artificial Explicable (XAI). A medida que los modelos de aprendizaje automático se vuelven más complejos, surge una preocupación crucial: ¿cómo podemos entender y confiar en las decisiones que estos modelos toman? La XAI se presenta como una respuesta fundamental a esta duda, al permitirnos desentrañar e indagar en el proceso de toma de decisiones de los modelos para brindar una explicación clara y comprensible para las elecciones que hacen.

La importancia de la Inteligencia Artificial Explicable radica en su capacidad para generar transparencia en los modelos de inteligencia artificial. A medida que estos modelos se aplican en áreas más complejas como la medicina, la justicia y el vehículo autónomo, se hace imprescindible que las decisiones que toman sean comprensibles tanto para aquellos que los desarrollan como a los usuarios finales. La XAI no solo mejora la confianza en la tecnología, sino que también abre la puerta a las implicaciones éticas y sociales de su implementación.

Para lograr la explicabilidad en los modelos de inteligencia artificial, se han desarrollado varios enfoques generales. Uno de ellos es la visualización de características relevantes. Tal y como se ha comentado antes en el modelo de Random Forest, esta técnica busca resaltar las características o atributos específicos de los datos que tienen un impacto significativo en las predicciones del modelo. Al identificar visualmente qué factores influyen más en las decisiones, surge una conexión directa entre la complejidad del algoritmo y la comprensión humana.

Otro enfoque es la interpretación de modelos internos. Esta metodología implica indagar en el funcionamiento interno del modelo y analizar sus componentes fundamentales. Por ejemplo, en la Red Neuronal Convolucional, esto implica comprender cómo las neuronas en diferentes capas interactúan para producir una predicción. En el caso de los Modelos de Random Forest, se podrían examinar el comportamiento y la toma de decisiones de los árboles individuales y su contribución a la predicción global.

En resumen, el objetivo de la Inteligencia Artificial Explicable es guiar y dar a entender todo aquello que desconocemos del comportamiento y la toma de decisiones de la inteligencia artificial. Su capacidad para aclarar el razonamiento de las máquinas y ofrecer transparencia es crucial para construir una sociedad que cada vez se sienta más cómoda y confíe en el uso de estas tecnologías, utilizándolas también de manera ética y responsable.

2.2.1 Explicabilidad de Modelos CNN

La Explicabilidad dentro del ámbito de los modelos de Redes Neuronales Convolucionales (CNN), al igual que con cualquier otro modelo, desempeña un papel crítico al buscar comprender y confiar en las decisiones tomadas por estos modelos de alta complejidad. La necesidad de comprender cómo se llega a ciertas predicciones y qué características de los datos influyen en estas decisiones se vuelve primordial, sobre todo en contextos de aplicaciones cruciales o con alta complejidad como diagnósticos médicos. En esta sección, entraremos en el análisis de las técnicas de explicabilidad, herramientas que permiten arrojar luz sobre el proceso interno de razonamiento de las CNN.

A pesar de que las CNN han demostrado su capacidad para lograr altos niveles de precisión en tareas de visión por computadora, las dudas en torno a su funcionamiento interno plantean cuestionamientos significativos. La incapacidad para justificar por qué un modelo toma una decisión particular puede dar lugar a preocupaciones éticas y de entendimiento, por ello se busca encontrar una justificación comprensible y transparente para sus predicciones.

Técnicas de Explicabilidad:

- **Mapas de Calor**: Una técnica en la interpretación de las CNN es la generación de mapas de calor. Estos mapas resaltan áreas en una imagen que tienen un mayor impacto en la predicción de la red. En caso de que los usuarios deseen comprender qué características específicas del objeto o la escena condujeron a la decisión del modelo, pueden consultar estos mapas para lograr un mayor entendimiento. Esta técnica ofrece una manera intuitiva de identificar qué partes de la entrada contribuyen más a la salida del modelo.

- **Análisis de Activación**: El análisis de activación implica el análisis de cómo se activan las neuronas en las distintas capas de una CNN al presentar una imagen como entrada. Visualizar las activaciones neuronales permite trazar la ruta de información mientras fluye a través de las capas. Esto ayuda a comprender qué características específicas la red está detectando en cada etapa y cómo estas contribuyen a las decisiones finales.

2.2.2 Explicabilidad de Modelos RF

La explicabilidad en el contexto de los Modelos de Random Forest (RF) , al igual que con las Redes Neuronales Convolucionales, es un elemento esencial para desentrañar el proceso de toma de decisiones basado en la combinación de múltiples árboles de decisión. Los Modelos de Random Forest son conocidos por la habilidad para combinar las predicciones de múltiples árboles de decisión en una predicción final más precisa. Sin embargo, la cuestión de cómo se llega a esta predicción combinada sigue siendo la duda, y por tanto, de gran interés. Tratando de averiguar el proceso, las características o atributos son los más influyentes en las decisiones tomadas por el modelo.

Métodos de Explicabilidad en Modelos RF

- **Características**: Una técnica en este tipo de modelos es la evaluación de la importancia de las características. Esta métrica cuantifica el impacto de cada atributo en la precisión de las predicciones. Al comprender qué características tienen un mayor peso en las decisiones del modelo, se obtiene una visión sobre qué aspectos de los datos influyen de manera más significativa respecto a otras en los resultados.
- **Visualización de Árboles Individuales**: Dado que un modelo de RF está compuesto por una colección de árboles de decisión individuales, la visualización de árboles individuales es una herramienta efectiva para la explicabilidad. Al observar un árbol específico dentro del bosque, se puede entender cómo opera de manera aislada y qué reglas o condiciones sigue para llegar a sus predicciones. Esta técnica aporta claridad al proceso de toma de decisiones en cada árbol y cómo contribuye a la salida combinada del RF.

3. Estado del Arte

3.1 Trabajos relacionados

En esta sección, se llevará a cabo una revisión de la investigación previa relacionada con la generación de explicaciones y otros trabajos relacionados que utilizan los datos proporcionados por Google en QuickDraw. Uno de los proyectos en este contexto es el trabajo realizado por Jim Vallandingham titulado "How Long Does it Take to (Quick) Draw a Dog?". El objetivo principal de este proyecto es analizar y comprender el tiempo promedio que las personas dedican a realizar dibujos en QuickDraw. Esto se lleva a cabo al comparar esta métrica tanto entre categorías similares como en un contexto general. Aunque en el título indique únicamente el tiempo que se tarde en dibujar un perro, también se realizan estudios de otras categorías, como animales, insectos y formas geométricas.

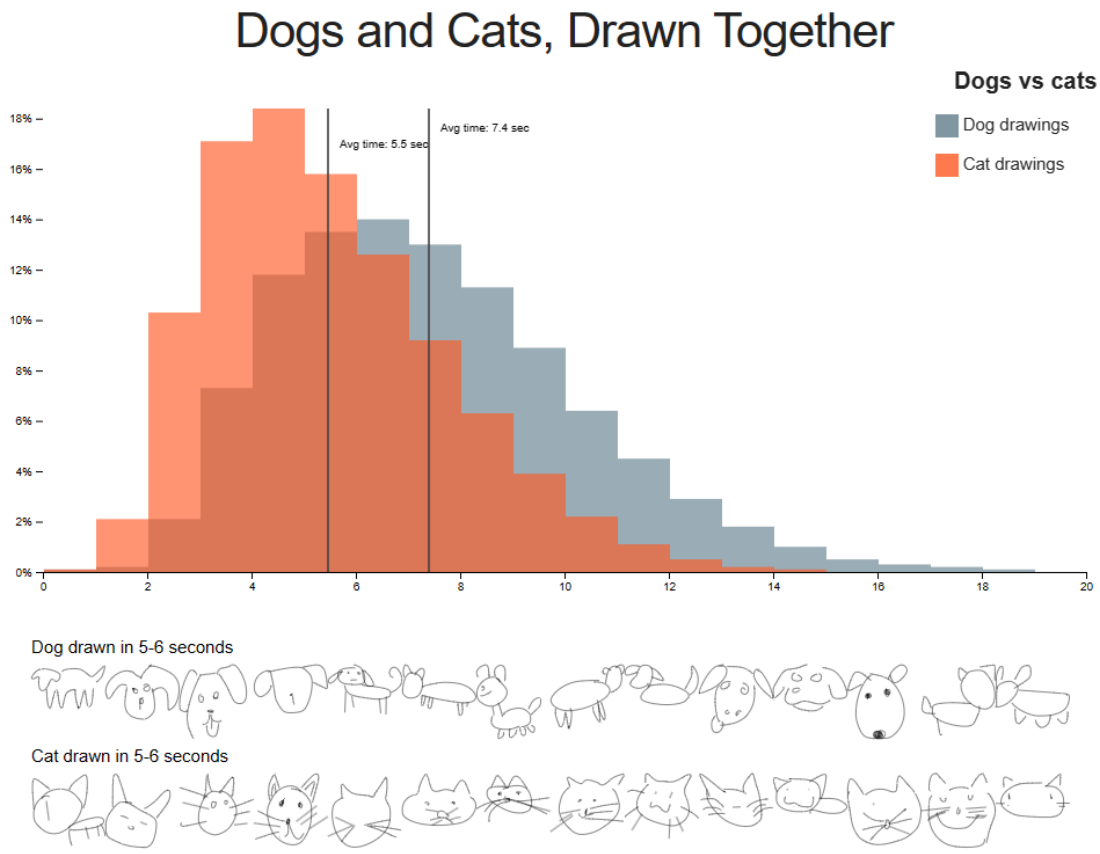


Figura 3. Proyecto "How Long Does it Take to (Quick) Draw a Dog?" [17]

Otro ejemplo de un proyecto que utiliza la base de datos de QuickDraw es el trabajo propuesto por Keisuke Irie. En su investigación, se enfoca en el desarrollo de un modelo de Redes Neuronales Convolucionales (CNN). En este proyecto, se evalúa la precisión del modelo para reconocer imágenes, y también se investiga la procedencia geográfica de los usuarios que han contribuido con dibujos a sus respectivas categorías. Este proyecto busca conseguir tanto el reconocimiento de imágenes como la de su posible ubicación geográfica, lo que podría ofrecer información valiosa sobre cómo los diferentes factores pueden influir en el rendimiento y aprendizaje de los modelos de aprendizaje automático.

	image recognition	Country prediction
CNN model	90.2%	62.7%
XGBoost model	79.1%	43.8%

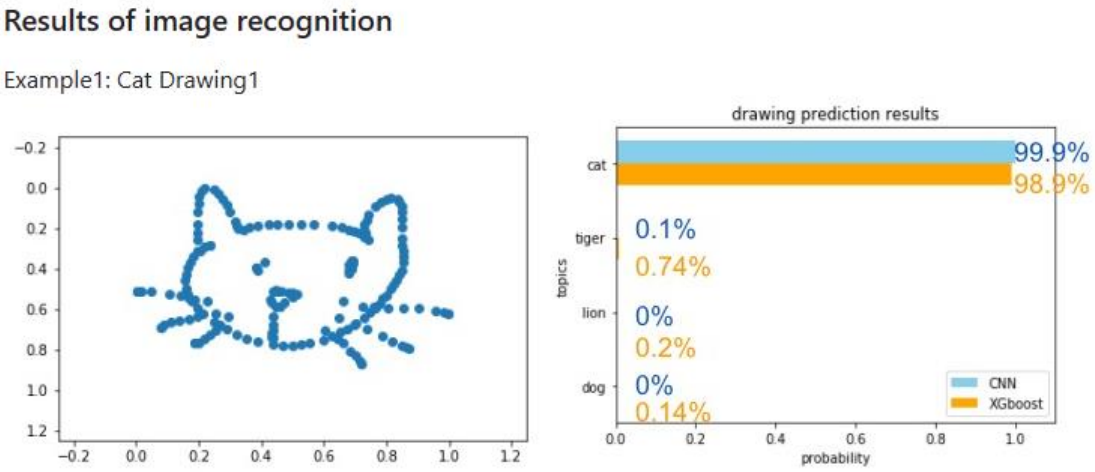


Figura 4. Proyecto "QuickDraw Prediction Model"[6]

Por último, encontramos una iniciativa por parte del gobierno de España, donde habla sobre cómo los datos abiertos pueden servir de entrenamiento para los modelos de aprendizaje automático. Los datos abiertos suelen ser más simples y menos ruidosos que los datos recopilados en entornos del mundo real, lo que puede facilitar el entrenamiento de los modelos de IA.

Figura 5. Logotipo de la Dirección General de Gobernanza Pública [22]

El artículo menciona varias iniciativas y organizaciones que están trabajando para hacer que los datos abiertos estén disponibles para el entrenamiento de modelos de machine learning. Una de estas iniciativas es “ML Commons”, que tiene como objetivo mejorar el impacto positivo del aprendizaje automático en la sociedad y acelerar la innovación ofreciendo herramientas como conjuntos de datos, mejores prácticas y algoritmos abiertos.[22]

3.2. Avances en explicabilidad en modelos de predicción de objetos en garabatos

La interpretación de garabatos y dibujos realizados a mano por parte de modelos de predicción de objetos representa un emocionante campo en la inteligencia artificial. A medida que estos modelos avanzan en su capacidad para identificar y etiquetar objetos en trazos sencillos, la cuestión de cómo llegan a esas conclusiones se vuelve primordial.

Los avances en la explicabilidad de modelos de predicción de garabatos se centran en desbloquear el razonamiento a la hora de etiquetar, es decir, en las clasificaciones. Dado que un simple garabato puede representar múltiples conceptos y objetos, comprender qué aspectos visuales influyen en la predicción es esencial.

Hay diversas técnicas de explicabilidad, por ejemplo, en cuanto a redes neuronales encontramos diversas técnicas posibles:

- **Deconvolución**: Este método se emplea con el propósito de visualizar las características que son aprendidas por las capas convolucionales dentro de una red neuronal. La idea es mapear las activaciones de la red de nuevo a la entrada del espacio de píxeles. Para hacer esto, se aplica una operación de “desenrollado” a las activaciones, que es esencialmente el proceso inverso de la convolución.[18] Esto permite visualizar qué características de la imagen de entrada son importantes para la predicción de la red.
- **Integrated Gradients**: Este método se utiliza para asignar la predicción de una red neuronal a sus entradas. Proporciona una forma de calcular las importancias de las características y es aplicable a cualquier modelo

que sea diferenciable. Integrated Gradients funciona calculando el gradiente del modelo con respecto a cada característica de entrada y luego integrando estos gradientes a lo largo del camino desde un punto de referencia hasta el punto de interés.[19]

- **Layer-wise Relevance Propagation (LRP)**: LRP es un método que permite descomponer la salida de la red en términos de relevancia sobre las neuronas de entrada. Esto se hace calculando cómo cada neurona contribuye a la salida final y luego propagando esta relevancia hacia atrás a través de la red hasta llegar a las neuronas de entrada.[1]

4. Conjunto de Datos

4.1 Descripción del Conjunto de Datos Quickdraw

El conjunto de datos de "Quick, Draw!", es una colección de datos creada por Google que terminó haciéndose pública. Este conjunto de datos se obtuvo a través de un juego en línea del mismo nombre que trata de dibujar objetos dentro de un tiempo limitado por el propio juego, normalmente 20 segundos. Los dibujos dibujados por los usuarios se recopilan para almacenarlo en la base de datos, pudiendo aprender el propio modelo de dichas imágenes o realizar investigaciones y análisis mediante dicho conjunto de datos, hecho que ocurrió tras la publicación de los datos.

Figura 6. Imágenes de dibujos de Quick, Draw! [4]

Los dibujos almacenados en el conjunto de datos se encuentran en formato vectorial etiquetados con metadatos, indicando el jugador, el país, la fecha y la categoría de este. El vector almacenado presenta secuencias de trazos o puntos que representan los dibujos realizados por los participantes. El conjunto de datos

abarca una amplia gama de categorías de objetos, incluyendo vehículos, objetos del día a día, animales, comida y más. La cantidad de dibujos puede variar significativamente de una categoría a otra, dependiendo la categoría podemos encontrar con cientos de miles de ejemplos, mientras que otras categorías pueden llegar a tener un poco menos, y la calidad y estilo de las imágenes también pueden variar, dado que son realizados por personas de todo el mundo que pueden poseer mayor o menor experiencia dibujando, por lo que podemos encontrar desde dibujos muy detallados y precisos a otros que son más sencillos o abstractos.

A parte, Google te permite extraer los datos del conjunto en diversos formatos. Pudiendo extraerlos el formato más básico y completo, que contiene todos los datos citados en un JSON, hasta incluso otros datos ya previamente procesados que se centran por ejemplo únicamente en la imagen del dibujo.

4.2 Parejas seleccionadas del conjunto de datos

En el proceso de entrenamiento y análisis de los modelos, se ha realizado una selección de cinco parejas cuyas categorías resultan similares, es decir, que ambos objetos son similares dentro de la pareja. Estas selecciones han basado en una evaluación subjetiva de similitud con el objetivo de comprender cómo los modelos son capaces de reconocer categorías que presentan similitudes visuales y estructurales, así como su habilidad para identificar categorías incluso cuando los dibujos son simples.

Las parejas de categorías elegidas son las siguientes: coche-camión, abeja-mariposa, arbusto-árbol, gato-perro y silla-mesa. Cada uno de estos conjuntos de categorías se utilizará en un proceso de entrenamiento y análisis separado. Esta división permite evaluar el rendimiento de los modelos de manera específica en cada pareja, lo que arrojará luz sobre cómo se desenvuelven frente a diferentes grados de similitud y complejidad en las imágenes.

Este enfoque de selección de parejas similares no solo contribuye a una comprensión más profunda de la capacidad de reconocimiento de las máquinas, sino que también ofrece información valiosa sobre cómo las categorías visuales que comparten rasgos distintivos pueden influir en el proceso de clasificación.

Estas decisiones de los modelos son fundamentales para abordar los objetivos de este trabajo de investigación y generar resultados significativos y relevantes.

4.3 Preparación del Conjunto de Datos para CNN y RF

En este apartado, se va a proporcionar una visión exhaustiva de los procedimientos que se han llevado a cabo para preparar el conjunto de datos antes de su utilización tanto en el modelo de Red Neuronal Convolucional (CNN) como en el Modelo de Random Forest (RF). Se va a detallar tanto el conjunto de datos utilizado para cada modelo como los pasos de limpieza y preparación utilizados.

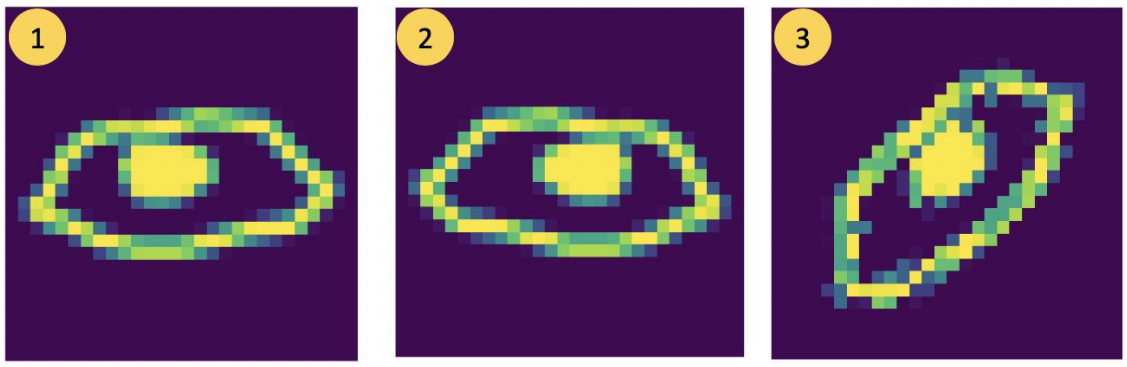


Figure 3 Image Preprocessing: (1) - original image, (2) - flipped image, (3) - rotated image.

Figura 7. Imágenes rotadas y reflejadas [7]

Para comenzar, se describe la preparación de datos realizada para el conjunto de datos empleado en el modelo de Red Neuronal Convolucional. En este caso, no se ha llevado a cabo una limpieza previa de los datos, sino que se han utilizado todos los datos disponibles en la base de datos de las parejas de categorías seleccionadas. Además, se han agregado las mismas imágenes rotadas y reflejadas para aumentar el número de imágenes de entrenamiento, permitiendo al modelo reconocer imágenes de las categorías independientemente de su orientación. La preparación de estos datos ha consistido en extraer el vector de trazos que el usuario ha dibujado para cada imagen de la base de datos de Google, este vector se etiqueta posteriormente con la categoría a la que pertenece y se pasa al siguiente paso.

Dado que el modelo de CNN requiere que los datos de entrada sean tensores en lugar de vectores de trazos se realiza la transformación de los vectores etiquetados anteriormente. Para lograr esto, se utiliza una función que transforma el vector de trazos en una imagen rasterizada de tamaño específico.

```
Procedimiento vector_a_raster(vector_imagenes, lado=28, diametro_linea=16, relleno=16, color_fondo=(0,0,0), color_trazo=(1,1,1)):
    """
    El relleno y el diámetro de la línea son relativos a la imagen original de 256x256.
    """

    lado_original = 256.

    superficie = CrearSuperficieImagen(Formato=ARGB32, Ancho=lado, Alto=lado)
    contexto = CrearContexto(superficie)
    ConfigurarSuavizado(contexto, MejorSuavizado)
    ConfigurarPuntaLinea(contexto, PuntaRedonda)
    ConfigurarUnionLinea(contexto, UnionRedonda)
    ConfigurarAnchoLinea(contexto, diametro_linea)

    # Escalar para que coincida con el nuevo tamaño
    # Agregar relleno en los bordes para el diámetro de la línea
    # y agregar relleno adicional para la suavización de bordes
    relleno_total = relleno * 2. + diametro_linea
    nueva_escala = lado / (lado_original + relleno_total)
    EscalarContexto(contexto, nueva_escala, nueva_escala)
    TraducirContexto(contexto, relleno_total / 2., relleno_total / 2.)

    imagenes_raster = []
    Para cada imagen_vector en vector_imagenes:
        # Limpiar el fondo
        ConfigurarFuenteRGB(contexto, color_fondo)
        Pintar(contexto)

        caja = Maximo(vector_imagen).en_el_eje(1)
        desplazamiento = ((lado_original, lado_original) - caja) / 2.
        desplazamiento = Redefinir(desplazamiento, -1, 1)
        centrado = [trazo + desplazamiento para trazo en imagen_vector]

        # Dibujar trazos, esta es la parte más intensiva en CPU
        ConfigurarFuenteRGB(contexto, color_trazo)
        Para cada x_vector, y_vector en centrado:
            MoverA(contexto, x_vector[0], y_vector[0])
            Para cada x, y en zip(x_vector, y_vector):
                LineaA(contexto, x, y)
            Trazo(contexto)

        datos = ObtenerDatos(superficie)
        imagen_raster = Copiar(ComoArray(datos)[::4])
        imagenes_raster.Agregar(imagen_raster)

    Devolver ComoArray(imagenes_raster).ComoTipo('float32') / 255.0
```

Figura 8. Pseudocódigo de función de rasterización de imagen

Esta transformación implica configurar parámetros como el diámetro, tamaño de la imagen y color. Luego, se escalan los datos y se convierten en una imagen. Finalmente, se normalizan los valores para que cada píxel de la imagen esté dentro del rango [0, 1]. Una vez que los datos se han transformado en imágenes raster, se convierten en tensores, que son el tipo de dato necesario para entrenar el modelo de CNN, práctica común en el procesamiento de imágenes con CNN.

Por otro lado, en el entrenamiento del modelo de Random Forest, a diferencia del modelo de Red Neuronal Convolucional, sí que se realiza una limpieza previa de los datos. Esto implica utilizar únicamente las imágenes que el primer modelo de CNN puede reconocer. Para lograr esto, el modelo de CNN se entrena con todos los datos y se prueba con el mismo conjunto para identificar imágenes que pueden ser reconocidas de manera confiable. Esta selección de imágenes se

realiza para garantizar que estén relacionadas con su categoría de manera precisa, ya que en el juego de QuickDraw, los usuarios pueden dibujar libremente sin estar necesariamente relacionados con la categoría indicada. Una vez seleccionadas estas imágenes, se someten al mismo proceso de preparación que el modelo anterior para que sean reconocidas por el modelo de Random Forest.

5. Metodología

5.1 Diseño Experimental

En esta sección, se detalla el diseño experimental usado para llevar a cabo nuestro estudio. Un buen diseño experimental es necesario para poder garantizar la calidad y la confiabilidad de los resultados, en este se describen los procedimientos de selección de características y metodología de evaluación utilizados en el experimento. En cuanto al procedimiento de recolección de datos, no se va a entrar mucho en detalle puesto que ya que en apartados anteriores se describe como se obtienen y de donde provienen los datos utilizados.

A continuación, se va a detallar el proceso de selección de características utilizado para representar las imágenes de los dibujos proporcionados por los usuarios en el contexto de los modelos usados. La selección de características comprende la utilización de descriptores visuales, como características de textura o forma, que se extraen con el propósito de enriquecer la información de entrada de los modelos. En particular, se aplica un conjunto de transformaciones a las imágenes vectoriales, convirtiéndolas en imágenes raster, lo que facilita su procesamiento en los modelos. A continuación, se describen los parámetros clave de esta transformación:

- **side**: Este parámetro, cuyo valor por defecto va a ser 28, representa la longitud de los lados de la nueva imagen raster. La función redimensiona las imágenes vectoriales para que encajen en un cuadrado de este tamaño.[20]
- **line_diameter**: Con un valor por defecto de 16, este parámetro controla el grosor de las líneas en la imagen raster resultante. Permite ajustar el grosor de las líneas en función de este valor, lo que influye en la apariencia visual de las imágenes.[20]
- **padding**: El valor por defecto de 16 en este parámetro indica la cantidad de relleno que se agrega alrededor de la imagen vectorial antes de convertirla en raster. Este parámetro resulta esencial para garantizar que las líneas en el borde de la imagen no se corten durante el proceso de conversión.[20]

- **bg_color** y **fg_color**: Estos dos parámetros, con valores por defecto de (0, 0, 0) y (1, 1, 1) respectivamente, controlan el color de fondo (bg_color) y el color de las líneas dibujadas (fg_color) en la imagen raster generada.[20]

En resumen, la función aplicada a las imágenes vectoriales realiza diversas transformaciones, que incluyen la redimensión, el ajuste del grosor de las líneas, el relleno y la modificación de los colores de fondo y de las líneas. Estas transformaciones son fundamentales para convertir las imágenes vectoriales en imágenes raster que puedan ser procesadas de manera efectiva por los modelos utilizados en este TFG.

5.1.1 Metodología de Evaluación

La evaluación de los modelos varía según el enfoque utilizado. Para el modelo de Redes Neuronales Convolucionales (CNN), se ha utilizado el conjunto de datos proporcionado por Quickdraw, el cual poseerá todas las imágenes vectoriales de las parejas de categorías seleccionadas. La evaluación de este modelo se realiza considerando dos aspectos principales.

En primer lugar, se supervisa el proceso de entrenamiento de la CNN utilizando la métrica de pérdida o “loss”. Esta métrica permite cuantificar la diferencia entre las predicciones del modelo y las etiquetas reales del conjunto de entrenamiento, donde un valor de pérdida más bajo indica un mejor rendimiento del modelo en el aprendizaje de los patrones presentes en los datos.

En segundo lugar, se evalúa el rendimiento de la CNN después de finalizar el entrenamiento. Para ello, se mide la capacidad del modelo para identificar imágenes correctamente. Esta evaluación se lleva a cabo mediante el conteo del número de imágenes reconocidas de manera precisa de todo el conjunto de datos.

Por otro lado, en el caso del modelo de Bosque Aleatorio (RF), se utiliza el conjunto de imágenes previamente identificadas por la CNN como conjunto de entrenamiento. Luego, se evalúa la precisión o “accuracy” del modelo RF en la tarea de identificar imágenes. Esta precisión se determina mediante pruebas independientes, donde se varía el número de imágenes utilizadas para el

entrenamiento del modelo RF, pasando de 10 imágenes a 30.000 imágenes. Este enfoque permite comprender cómo el modelo RF trabaja en la identificación de imágenes de garabatos dependiendo del número de imágenes utilizadas para su entrenamiento.

En resumen, la metodología de evaluación adapta su enfoque dependiendo del modelo en consideración. En el caso de la CNN, se tiene en cuenta tanto el proceso de entrenamiento como el rendimiento final, utilizando el conjunto de datos proveniente de Quickdraw. Para el modelo RF, se utilizan las imágenes reconocidas por el modelo de CNN para evaluar su precisión en la tarea de identificación de imágenes de garabatos. Esta combinación de enfoques asegura una evaluación exhaustiva de los modelos en un contexto relevante y desafiante.

5.1.2 Variables Controladas

En el proceso de diseño experimental, es fundamental identificar y controlar cualquier variable que pueda influir en los resultados del estudio. Estas variables controladas son elementos que se mantienen constantes o se manipulan de manera específica para asegurar la consistencia y la confiabilidad de las pruebas. En este estudio, se han identificado las siguientes variables controladas:

- **Arquitectura del Modelo**: Tanto para el modelo de Convolutional Neural Network (CNN) como para el modelo de Random Forest (RF), se ha establecido previamente una arquitectura específica que define la estructura y los componentes clave de cada modelo. Esta arquitectura se ha diseñado de manera coherente y se ha mantenido constante a lo largo de las pruebas, lo que garantiza que los resultados sean comparables y que cualquier diferencia observada se deba a otros factores.
- **Conjunto de Datos de Entrada en CNN**: En el caso del modelo de CNN, el conjunto de datos de entrada se ha establecido como las imágenes extraídas de Quickdraw de categorías específicas. Este conjunto de datos también se ha mantenido constante para todas las pruebas relacionadas con el modelo de CNN, lo que ayuda a evaluar la efectividad del modelo en la clasificación de estas imágenes.

- **Tamaño del Conjunto de Datos de entrenamiento para RF**: Para el modelo de RF, se ha controlado el tamaño del conjunto de datos de entrada utilizado para el entrenamiento. Este tamaño varía de 10 a 30,000 imágenes, lo que permite evaluar cómo la cantidad de datos de entrenamiento afecta a la precisión del modelo. Además, se ha aplicado un ordenamiento previo a las imágenes por el número de trazos, asegurando que el modelo de RF siempre utilice primero las imágenes más simples y con menos trazos durante el entrenamiento.

El control de estas variables garantiza que las pruebas se realicen de manera coherente y que cualquier cambio en los resultados pueda atribuirse a otros factores, como la cantidad de datos de entrenamiento o las características específicas de la arquitectura de los modelos. Esto contribuye a la validez y la confiabilidad de las conclusiones obtenidas en este estudio.

5.2 Arquitectura y Entrenamiento del Modelo CNN

En esta sección, se presenta en detalle la arquitectura del modelo de Redes Neuronales Convolucionales (CNN) destacando que la elección de esta arquitectura fue considerada para la tarea en cuestión.

Esta red CNN toma imágenes de entrada en escala de grises de 28x28 píxeles y las procesa a través de una capa de convolución, una capa de “MaxPooling” y dos capas completamente conectadas. La capa de convolución extrae características de las imágenes, y las capas completamente conectadas realizan la clasificación final. Cada parámetro de la arquitectura es configurable, lo que permite adaptar el modelo a diferentes tareas y requisitos. Esta arquitectura predefinida sirve como una base sólida para el aprendizaje supervisado y la generación de explicaciones en el contexto de clasificación de objetos en garabatos.

```
Procedimiento crearModeloCNN(X_train, y_train, X_test, y_test):
    # Parsear argumentos de línea de comandos
    learning_rate = 0.001
    epochs = 30
    weight_decay = 0
    dropout = 0.0
    mini_batches = 1000
    optimizer = 'SGD'
    gpu = Falso

    learning_rate = ObtenerArgumentoFlotante('--learning_rate', 0.001)
    epochs = ObtenerArgumentoEntero('--epochs', 30)
    weight_decay = ObtenerArgumentoFlotante('--weight_decay', 0)
    dropout = ObtenerArgumentoFlotante('--dropout', 0.0)
    mini_batches = ObtenerArgumentoEntero('--mini_batches', 1000)
    optimizer = ObtenerArgumento('--optimizer', 'SGD', opciones=['SGD', 'Adam'])
    gpu = ObtenerArgumentoBooleano('--gpu')

    resultado = ParsearArgumentos()

    arquitectura = resultado.arquitectura
    n_chunks = 824

    si gpu == Verdadero entonces
        dispositivo = 'cuda'
    sino
        dispositivo = 'cpu'

    si resultado.save_dir == ' ' entonces
        ruta_guardado = 'checkpoint.pth'
    sino
        ruta_guardado = resultado.save_dir + '/' + 'checkpoint.pth'

    # Agregar imágenes reflejadas y rotadas al conjunto de datos
    si resultado.add_data == Verdadero entonces
        X_train, y_train = agregar_imagenes_reflejadas_y_rotadas(X_train, y_train)

    # Guardar conjuntos de datos en disco si es necesario
    guardar_datos(X_train, y_train, X_test, y_test, forzar = resultado.add_data)

    # Convertir a tensores
    entrenamiento = ConvertirATensor(X_train).Flotante()
    etiquetas = ConvertirATensor(y_train).Entero()
    prueba = ConvertirATensor(X_test).Flotante()
    etiquetas_prueba = ConvertirATensor(y_test).Entero()

    # Hiperparámetros para nuestra red
    tamaño_entrada = 784
    tamaños_ocultos = [128, 100, 64]
    tamaño_salida = 2

    # Construir modelo
    modelo = construir_modelo(tamaño_entrada, tamaño_salida, tamaños_ocultos,
                              arquitectura = arquitectura, dropout = dropout)

    # Ajustar modelo
    ajustar_modelo(modelo, entrenamiento, etiquetas, epochs = epochs, n_chunks = n_chunks,
                   learning_rate = learning_rate, weight_decay = weight_decay, optimizer = optimizer)

    Devolver modelo
```

Figura 9. Pseudocódigo de construcción del modelo CNN

A continuación, se detallan los componentes clave de la arquitectura:

Capa de convolución 1 (conv1):

- Tipo: Capa de convolución (nn.Conv2d)
- Número de canales de entrada: 1 (indicando imágenes en escala de grises)
- Número de canales de salida: 18
- Tamaño del kernel: 3x3

- Stride: 1
- Padding: 1
- Función de activación: ReLU (F.relu)

Esta capa inicial de convolución es esencial para extraer características relevantes de las imágenes de entrada. El uso de 18 canales de salida permite la detección de múltiples características y patrones en las imágenes.[16]

Capa de MaxPooling (pool):

- Tipo: Capa de MaxPooling (nn.MaxPool2d)
- Tamaño del kernel: 2x2
- Stride: 2
- Padding: 0

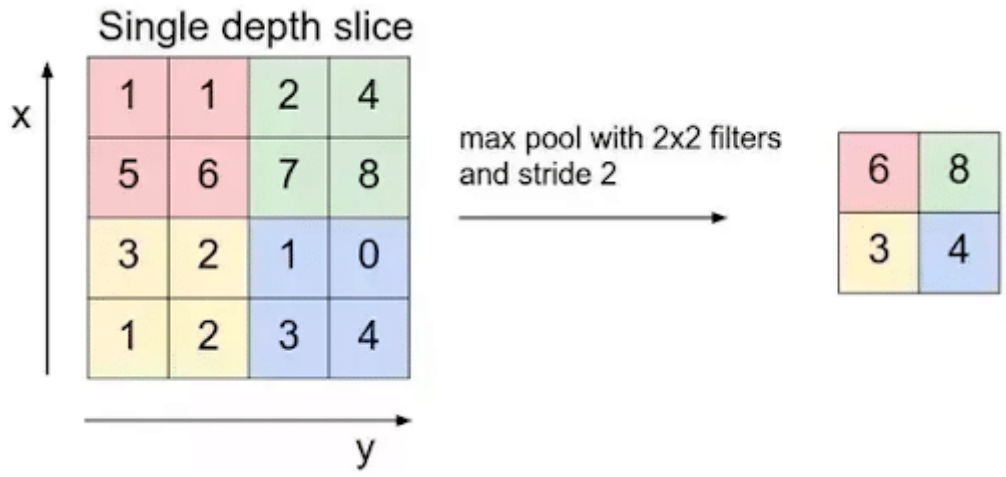


Figura 10. Funcionamiento de la capa MaxPooling[16]

La capa de MaxPooling se utiliza para reducir la dimensionalidad de las características extraídas, conservando las características más importantes. Esto ayuda a disminuir la complejidad del modelo y reduce el riesgo de sobreajuste.

Capa completamente conectada 1 (fc1):

- Tipo: Capa completamente conectada (nn.Linear)
- Número de nodos de entrada: 18 * 14 * 14 (resultado de la capa de convolución y el MaxPooling)
- Número de nodos de salida: 2
- Función de activación: ReLU (F.relu)

La primera capa completamente conectada se encarga de combinar las características extraídas en una representación adecuada para la clasificación. El uso de la función de activación ReLU ayuda a introducir no linealidades en el modelo, lo que puede ser crucial para aprender relaciones complejas.

Capa completamente conectada 2 (fc2):

- Tipo: Capa completamente conectada (nn.Linear)
- Número de nodos de entrada: [128, 100, 64]
- Número de nodos de salida: 2 (2 categorias)

La segunda capa completamente conectada produce las salidas finales del modelo, correspondientes a las clases de clasificación. El número de nodos de salida es igual al número de clases y es configurable según la tarea específica.

5.3 Construcción y Entrenamiento del Modelo RF

Esta sección se enfoca en la construcción y entrenamiento del Modelo de Random Forest (RF). La elección de esta arquitectura específica se basó en consideraciones clave relacionadas con la naturaleza de la tarea y los requisitos de rendimiento. A continuación, se describen los parámetros utilizados en el modelo RF junto con las justificaciones detrás de su selección.

En cuanto al pseudocódigo de la construcción del modelo, este ya viene predefinido por la clase *RandomForestClassifier* del módulo de *sklearn.ensemble* [13], con lo no requiere de ningún calculo ni configuración previa a diferencia del modelo de CNN, quedando de la siguiente forma.

```
Procedimiento random_forest_classifier(X_train, y_train):
    # Instanciar el clasificador de bosque aleatorio
    modelo = ClasificadorBosqueAleatorio(n_estimators=100, max_depth=None)

    # Ajustar el clasificador
    AjustarModelo(modelo, X_train, y_train)

    Devolver modelo
```

Figura 11. Pseudocódigo de construcción del modelo RF

Parámetros del Modelo RF:

- n_estimators: 100 (número de árboles en el bosque)
- criterion: "gini" (criterio para medir la calidad de la división)

- max_depth: None (profundidad máxima de los árboles)
- min_samples_split: 2 (número mínimo de muestras para dividir un nodo)
- min_samples_leaf: 1 (número mínimo de muestras en una hoja)
- min_weight_fraction_leaf: 0.0 (fracción mínima del peso total de las muestras para estar en una hoja)
- max_features: "sqrt" (número de características consideradas para la mejor división)
- max_leaf_nodes: None (máximo número de nodos hoja)
- min_impurity_decrease: 0.0 (umbral mínimo de disminución de impureza para dividir un nodo)
- bootstrap: True (si se deben utilizar muestras de arranque al ajustar cada árbol)
- oob_score: False (si se deben usar muestras fuera de la bolsa para estimar la puntuación de generalización)
- n_jobs: None (número de trabajadores para paralelizar el entrenamiento y la predicción)
- random_state: None (controla la aleatoriedad en la construcción de árboles)
- verbose: 0 (controla la verbosidad al ajustar y predecir)
- warm_start: False (si se debe reutilizar la solución del ajuste anterior al agregar más estimadores)
- class_weight: None (pesos asociados a las clases para abordar el desequilibrio de clases)
- ccp_alpha: 0.0 (parámetro de complejidad para la poda de coste mínimo)
- max_samples: None (número de muestras para entrenar cada estimador base)

La elección de esta configuración de parámetros se basa en la naturaleza de la tarea y los objetivos del estudio. El número de estimadores (n_estimators) se establece en 100 para equilibrar el compromiso entre rendimiento y tiempo de entrenamiento.[12] El criterio “gini” se utiliza para medir la calidad de la división de nodos, lo que es adecuado para problemas de clasificación.

La profundidad máxima de los árboles (max_depth) se mantiene como “None” para permitir que los árboles crezcan hasta su máxima profundidad posible y capturen relaciones complejas en los datos. Otros parámetros, como “min_samples_split”, “min_samples_leaf” y “min_impurity_decrease”, se ajustan para evitar el sobreajuste y controlar la estructura de los árboles.

La selección de “sqrt” como “max_features” asegura que se considere un número razonable de características en cada división, lo que es útil en problemas con un gran número de características.[12] Se habilita el muestreo de arranque (bootstrap) para mejorar la generalización del modelo, y se desactiva la estimación “oob_score” para simplificar el proceso.

En resumen, la elección de estos parámetros se realiza cuidadosamente para optimizar el rendimiento del Modelo RF en la tarea específica de clasificación de objetos en garabatos, asegurando un equilibrio adecuado entre la complejidad del modelo y su capacidad de generalización.

6. Resultados

6.1 Generación de Resultados con el Modelo CNN

En esta sección, se presenta la generación de resultados del modelo de Red Neuronal Convolucional (CNN) utilizado en el estudio. Para la evaluación del modelo, se llevan a cabo dos análisis que proporcionan información sobre su rendimiento y capacidad de generalización.

El primer análisis se centra en el progreso del entrenamiento del modelo CNN. Durante el proceso de entrenamiento, es fundamental supervisar cómo evoluciona el rendimiento del modelo a medida que se expone a los datos de entrenamiento. Para esto, se calcula la pérdida del modelo en cada época del entrenamiento, utilizando en nuestro caso 30 épocas para el entrenamiento del modelo. La pérdida es una métrica que representa la diferencia entre las salidas predichas por el modelo y las salidas reales en el conjunto de entrenamiento. Un descenso continuo en la pérdida indica que el modelo está aprendiendo y ajustándose adecuadamente a los datos. Este análisis proporciona una visión detallada de la convergencia del modelo y su capacidad para aprender patrones en los datos de entrada.

```
Procedimiento probarModeloCNN(modeloCNN):
    # Poner el modelo en modo de evaluación
    modeloCNN.eval()

    dibujos_lista_auto = []
    dibujos_lista_camion = []

    # Ordenar por simplicidad
    desempacar_auto = ordenar(desempacar_dibujos('google_data/p1/full_binary_car.bin'), clave=lambda x: x['n_strokes'])
    para cada dibujo en desempacar_auto:
        # Agregar el dibujo a la lista
        agregarAlista(dibujos_lista_auto, dibujo['image'])

    # Ordenar por simplicidad
    desempacar_camion = ordenar(desempacar_dibujos('google_data/p1/full_binary_truck.bin'), clave=lambda x: x['n_strokes'])
    para cada dibujo en desempacar_camion:
        # Agregar el dibujo a la lista
        agregarAlista(dibujos_lista_camion, dibujo['image'])

    raster_imagenes_auto = vector_a_raster(dibujos_lista_auto)
    raster_imagenes_camion = vector_a_raster(dibujos_lista_camion)
    vector_auto = []
    vector_camion = []

    para cada fila en raster_imagenes_auto:
        etiqueta, nombre_etiqueta, predicciones = obtener_prediccion_CNN(modeloCNN, fila)
        si etiqueta == 0 entonces
            agregarAlista(vector_auto, fila)

    para cada fila en raster_imagenes_camion:
        etiqueta, nombre_etiqueta, predicciones = obtener_prediccion_CNN(modeloCNN, fila)
        si etiqueta == 1 entonces
            agregarAlista(vector_camion, fila)

    Devolver vector_auto, vector_camion
```

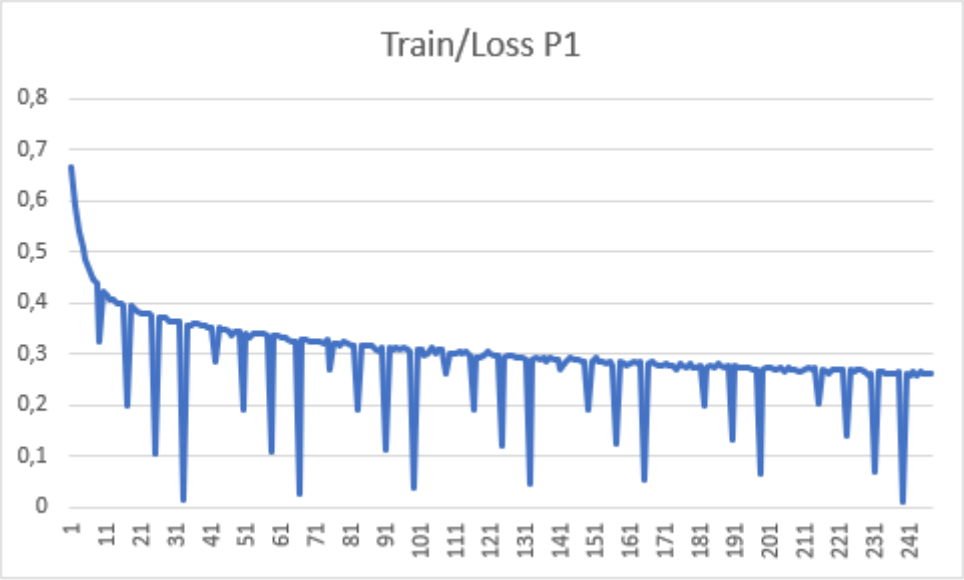
Figura 12. Pseudocódigo de la función de pruebas del modelo CNN

El segundo análisis se enfoca en evaluar la capacidad del modelo CNN para reconocer imágenes una vez que ha completado su entrenamiento. Se utiliza un conjunto de datos de prueba independiente que no se utilizó durante el entrenamiento ni la validación. El modelo realiza predicciones en este conjunto de datos y se calcula la tasa de reconocimiento, es decir, la proporción de imágenes que el modelo ha clasificado correctamente. Este análisis proporciona una medida fundamental de la capacidad del modelo para generalizar y aplicar sus conocimientos a nuevas imágenes.

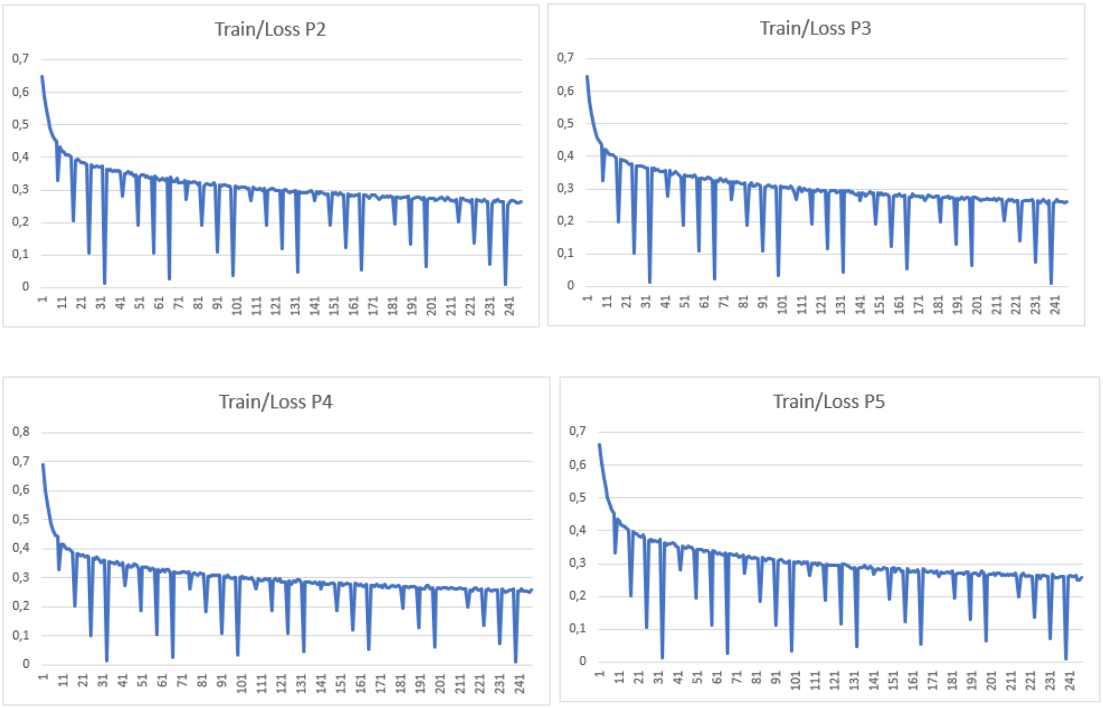
Ambos análisis se realizan para evaluar el rendimiento y la eficacia del modelo CNN en la tarea de clasificación de objetos en garabatos. Los resultados obtenidos en estos análisis proporcionan información esencial para comprender la capacidad de aprendizaje y generalización del modelo, lo que contribuye a una evaluación completa de su desempeño en la tarea específica.

6.2 Análisis de Resultados de la CNN

A continuación se van a mostrar los resultados obtenidos del modelo de CNN.



Gráfica 1. Resultado de pérdidas del modelo CNN de la pareja 1



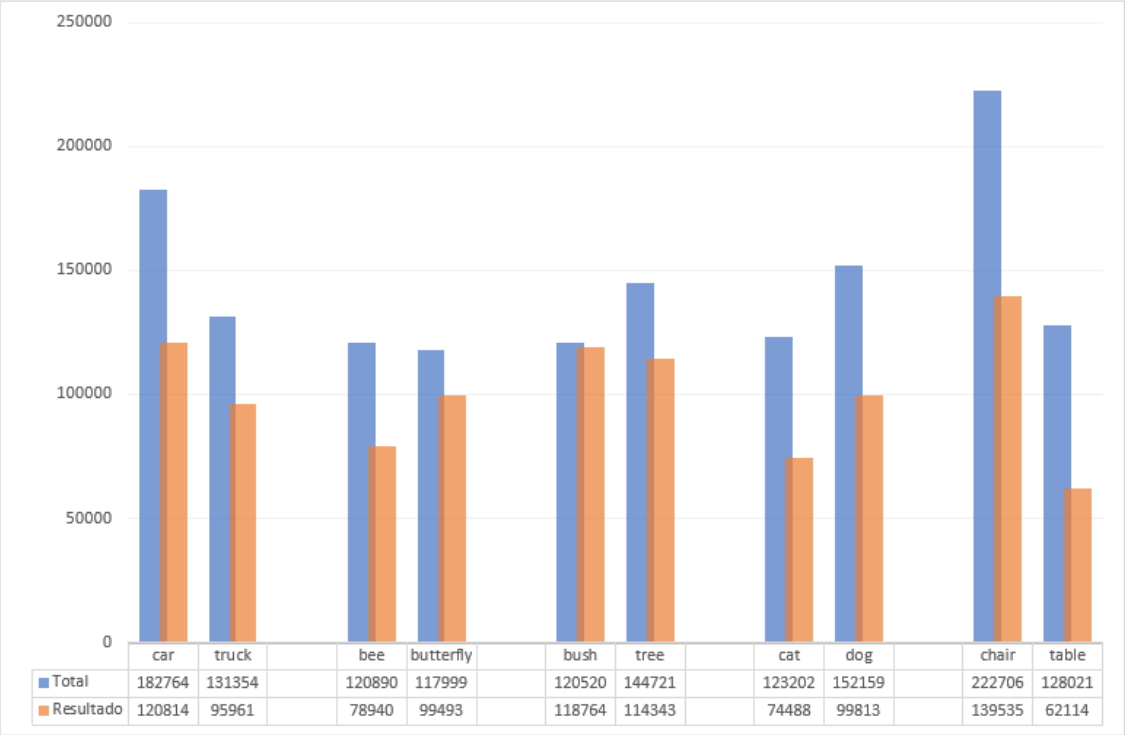
Gráfica 2. Resultado de pérdidas del modelo CNN de las parejas 2, 3, 4, 5

A primera vista podemos ver como los valores de pérdida disminuyen a lo largo del tiempo, lo que sugiere que el modelo está convergiendo hacia un estado en el que se ajusta cada vez mejor a los datos de entrenamiento. Esto es un indicativo positivo de que el modelo está aprendiendo.

A parte, conforme llegamos al final de la gráfica, podemos observar que la estabilidad es evidente en la consistencia de los valores de pérdida. La variación entre los valores es relativamente baja, lo que sugiere que el modelo mantiene un comportamiento constante durante el entrenamiento.

También durante todo el recorrido de la gráfica, se pueden identificar épocas específicas donde ocurren cambios significativos en los valores de pérdida. Estos puntos pueden señalar momentos clave en el entrenamiento, como la convergencia inicial o la adaptación a patrones específicos en los datos.

Al consultar los resultados obtenidos en la segunda prueba, resulta interesante observar el número reconocidas por el modelo de CNN en comparación con el conjunto de datos almacenados en QuickDraw.



	car	truck	bee	butterfly	bush	tree	cat	dog	chair	table
Total	182764	131354	120890	117999	120520	144721	123202	152159	222706	128021
Resultado	120814	95961	78940	99493	118764	114343	74488	99813	139535	62114

Gráfica 3. Resultados de imágenes reconocidas por el modelo CNN

Tras observar los resultados, resulta interesante ver que el número de imágenes reconocidas por el modelo varía según las parejas de categorías utilizadas. Aunque en algunos casos, el modelo muestra una precisión más alta, mientras que en otros puede ser un poco menos preciso. A pesar de esta variación, es alentador ver que incluso en el escenario menos favorable, la precisión del modelo supera consistentemente el 50%.

Este resultado sugiere que el modelo de CNN posee una capacidad de aprendizaje sólida y es capaz de reconocer patrones relevantes en las diferentes categorías de imágenes. Sin embargo, también subraya la importancia de la elección cuidadosa de las categorías de imágenes, ya que algunas pueden resultar más desafiantes que otras y afectar el rendimiento general. Algunas categorías pueden ser naturalmente más desafiantes de distinguir que otras, lo que puede afectar la precisión general.

6.3 Generación de Resultados con el Modelo RF

Esta sección se centra en la generación de resultados y análisis del Modelo de Random Forest (RF) utilizado en el estudio. El análisis se divide en dos partes

fundamentales, cada una diseñada para evaluar diferentes aspectos del modelo y su capacidad de generalización.

La primera parte del análisis implica la utilización de imágenes previamente reconocidas por el Modelo de Red Neuronal Convolucional (CNN). Estas imágenes sirven como datos de entrada para el Modelo RF. Durante el proceso de entrenamiento del RF, se varía el número de imágenes utilizadas para el entrenamiento, comenzando con un conjunto pequeño de 10 imágenes y aumentándolo gradualmente hasta un valor máximo de 30.000 imágenes. Para cada configuración, se mide la precisión del modelo RF en un conjunto de datos de prueba independiente.

La métrica clave para evaluar el rendimiento del Modelo RF es la precisión. La precisión se calcula como la proporción de imágenes clasificadas correctamente en el conjunto de prueba en relación con el número total de imágenes en el conjunto de prueba. Al variar el número de imágenes utilizadas en el entrenamiento del RF, se puede observar cómo afecta a su capacidad para realizar predicciones precisas. Este análisis proporciona información sobre cómo la cantidad de datos de entrenamiento influye en el rendimiento del modelo RF.

```
Procedimiento evaluar_clasificador_bosque_aleatorio(modelo, X_train, y_train, X_test, y_test):
    # Obtener predicciones
    y_predicciones_entrenamiento = predecir(modelo, X_train)
    y_predicciones_prueba = predecir(modelo, X_test)

    # Calcular puntajes de precisión
    precision_entrenamiento = calcular_puntaje_precision(y_train, y_predicciones_entrenamiento)
    precision_prueba = calcular_puntaje_precision(y_test, y_predicciones_prueba)

    Devolver precision_entrenamiento, precision_prueba
```

Figura 13. Pseudocódigo de la función de pruebas del modelo RF

Este cálculo de precisión se realiza gracias a la clase *accuracy_score* de la librería *sklearn* [14], gracias a ella únicamente debemos de llamar a las funciones de la librería que nos interesan.

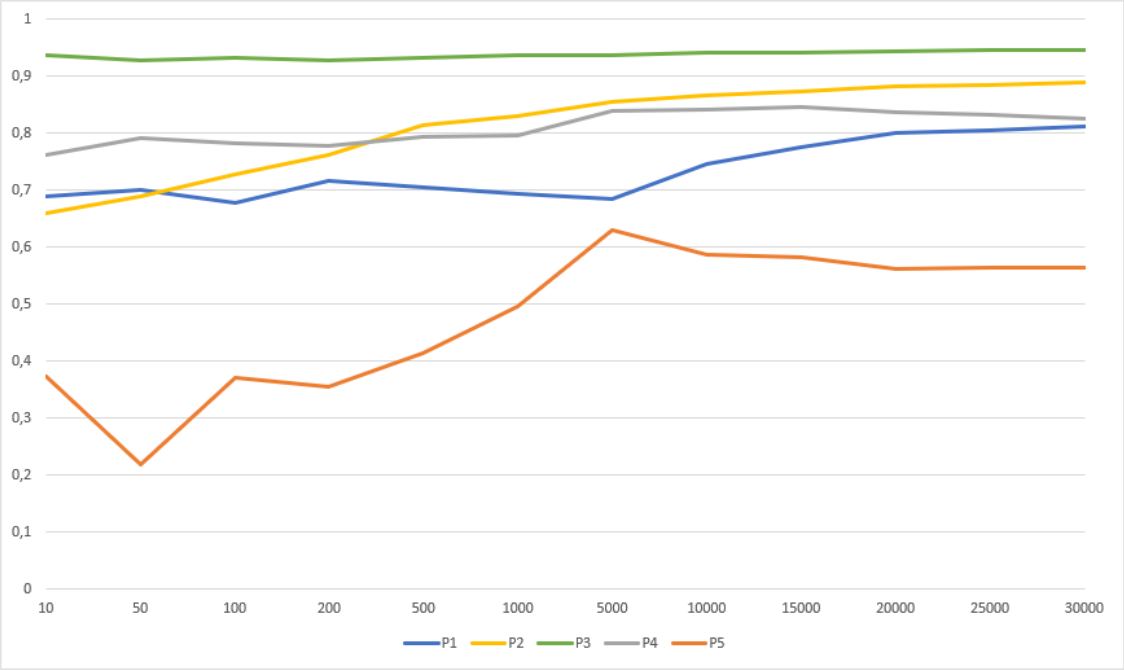
Este enfoque de análisis permite comprender la capacidad del Modelo RF para aprender de las imágenes reconocidas previamente por el CNN y cómo mejora su precisión a medida que se le suministra más información durante el entrenamiento. La evaluación de precisión en diferentes configuraciones de entrenamiento ofrece una visión detallada de cómo se comporta el modelo en términos de generalización y capacidad predictiva.

6.4 Análisis de Resultados del Modelo de RF

En esta sección, se llevará a cabo el análisis de los resultados obtenidos mediante el modelo de Random Forest (RF) implementado en este estudio. El modelo RF se someterá a una evaluación, teniendo en cuenta factores clave como la precisión, el desempeño en la identificación de categorías.

	P1	P2	P3	P4	P5
10	0,69025638	0,65941089	0,93691327	0,76277129	0,37291544
50	0,70076786	0,68936889	0,92825456	0,7925804	0,2176869
100	0,67746933	0,72923382	0,9336207	0,78206175	0,37136969
200	0,71702959	0,76201041	0,92856874	0,77707263	0,35572368
500	0,70462456	0,81369389	0,93355787	0,79358576	0,41340656
1000	0,69288807	0,82996358	0,93791864	0,79702914	0,49562037
5000	0,68603294	0,85484254	0,93828309	0,83966923	0,62937177
10000	0,74737892	0,86745643	0,94111068	0,84204441	0,58698302
15000	0,77574388	0,87441919	0,94303344	0,84709638	0,5831375
20000	0,79990662	0,8817866	0,94481797	0,83819889	0,56189914
25000	0,80651768	0,88496798	0,94617521	0,83166401	0,56462619
30000	0,81277856	0,88946098	0,94696694	0,82512913	0,56377163

Gráfica 4. Tabla de resultados de precisión en función de imágenes usadas en entrenamiento

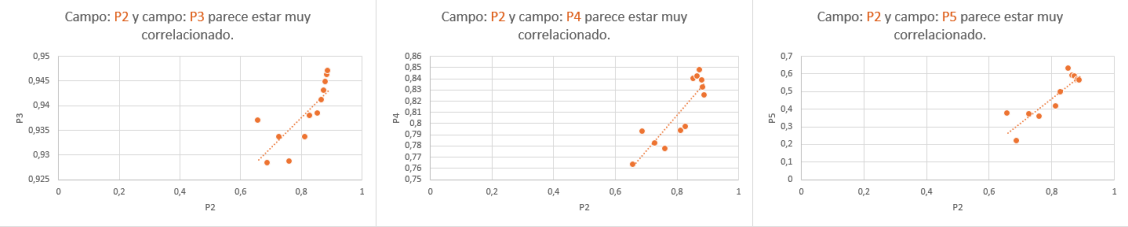


Gráfica 5. Gráfica de líneas de resultados de precisión

A simple vista, podemos apreciar un sólido desempeño en todos los modelos de P1 a P4. Durante el proceso de entrenamiento, se observa una tendencia general

al aumento en la precisión a medida que se incrementa el número de imágenes utilizadas en el entrenamiento. En todos los casos, los valores iniciales superan el umbral del 0.65 y, en la mayoría de los casos, alcanzan valores superiores al 0.8.

Sin embargo, se presenta una excepción en el caso de P5, al inicio del entrenamiento con una precisión significativamente inferior en comparación con los otros modelos. Esta discrepancia puede deberse a la estrategia de ordenar las imágenes desde las más simples hasta las más complejas. Este enfoque conlleva que el entrenamiento se realice con imágenes más simples, lo que, dependiendo de la categoría en cuestión, puede resultar en un desafío para el modelo, especialmente si la distinción entre categorías se vuelve menos evidente en dibujos más simples. Aunque tal y como vemos en P5, al uso de 5.000 imágenes para el entrenamiento, se observa un aumento considerable en la precisión.



Gráfica 6. Análisis de correlaciones entre parejas

Sin lugar a duda, el análisis de las correlaciones entre las parejas de categorías revela datos interesantes en el estudio. No obstante, cabe destacar que estas correlaciones no están intrínsecamente relacionadas entre sí. En otras palabras, cada pareja de categorías se evalúa por separado y por tanto las correlaciones entre ellas no nos proporcionan una comprensión directa de su interconexión.

Esta aparente falta de relación entre las parejas de categorías podría sugerir que, a pesar de pertenecer a grupos diferentes, algunas de ellas comparten características visuales o estructurales notables que influyen en su reconocimiento tanto por parte de los modelos de CNN como por los sujetos humanos. Estas características podrían ser la simplicidad en la representación visual, la presencia de rasgos distintivos o similitudes en la estructura general de las imágenes.

De esta manera, podemos considerar que las correlaciones detectadas entre las parejas de categorías nos brindan una visión más profunda de cómo los elementos visuales y estructurales impactan en el proceso de reconocimiento. Sin embargo, para comprender a fondo por qué algunas categorías son más fáciles de reconocer que otras, sería necesario llevar a cabo un estudio adicional que analice en detalle estas características compartidas y su influencia en el rendimiento de los modelos y los sujetos humanos.

6.5 Generación de Resultados para las pruebas con sujetos reales.

Esta sección se enfoca en la generación de imágenes usadas para la realización de pruebas con sujetos reales. Estas pruebas involucran la selección de imágenes del conjunto previamente reconocido por el Modelo de Red Neuronal Convolucional (CNN). Las imágenes seleccionadas representan cada una de las cinco parejas de categorías definidas en el estudio. Para una evaluación integral, se siguen tres enfoques distintos de selección de ejemplos explicativos:

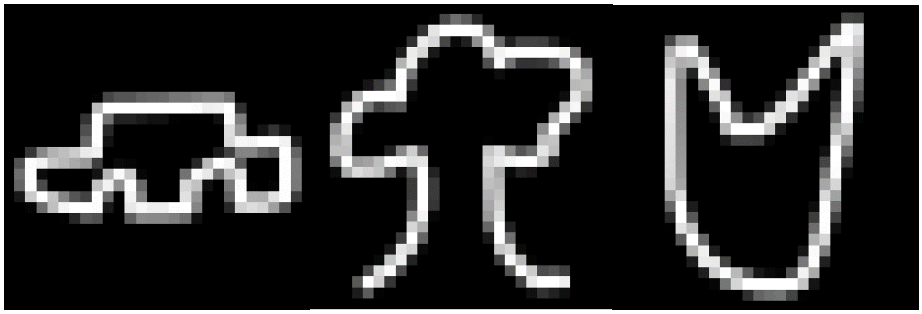


Figura 14. Ejemplo de imágenes más simples

En el primer enfoque, se seleccionan los cinco ejemplos más simples de cada categoría de cada pareja. Estos ejemplos se caracterizan por poseer trazos mínimos en sus representaciones, lo que los convierte en casos ideales para evaluar la capacidad del humano a reconocer imágenes simples o abstractas. Para seleccionar las imágenes únicamente se escogen las primeras imágenes obtenidas tras la limpieza del modelo CNN, se escogen las primeras imágenes dado que las imágenes vienen ordenadas en función de la simplicidad, de mayor a menor simplicidad.

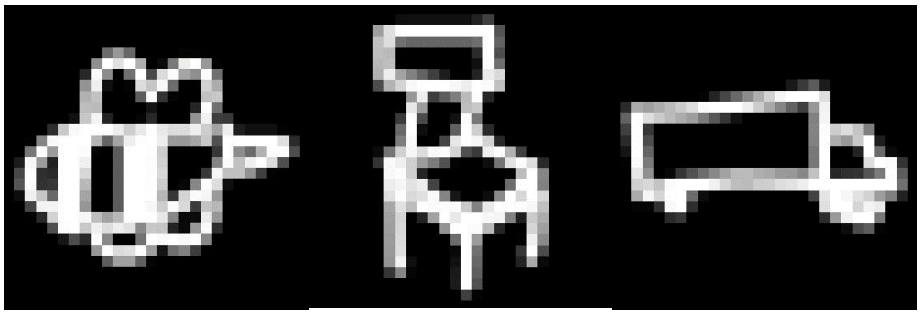


Figura 15. Ejemplo de imágenes random

El segundo enfoque implica la selección aleatoria de cinco ejemplos de cada categoría de cada pareja de categorías, también a partir del conjunto de imágenes reconocidas por el Modelo CNN. Esta elección aleatoria busca proporcionar una visión equilibrada de los resultados y evaluar la capacidad de reconocimiento del humano en un conjunto diverso de ejemplos.

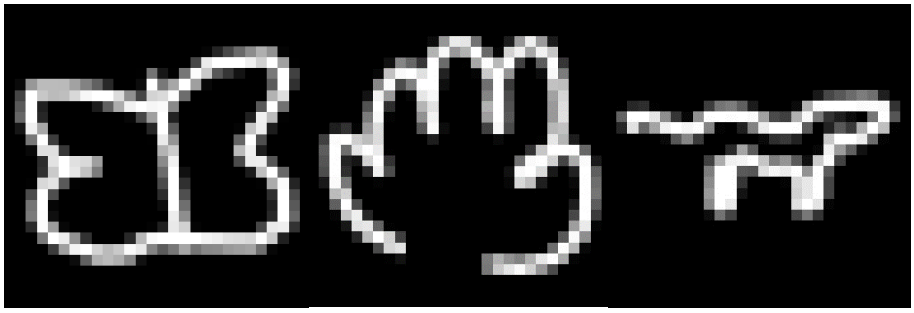


Figura 16. Ejemplo de imágenes simples con mayor certeza

En el tercer enfoque, se seleccionan los cinco ejemplos con mayor certeza dentro de aquellos que forman parte de los ejemplos más simples, los cuales forman parte del conjunto de imágenes reconocidas por el Modelo de Bosque Aleatorio (RF), teniendo en cuenta que . Estos ejemplos se consideran como aquellos en los que el modelo RF ha demostrado un alto grado de confianza en sus predicciones. El procedimiento para obtener los 5 ejemplos con mayor certeza sería el siguiente:

```
Procedimiento probarModeloRF(modeloRF, vector):
    predicciones_a_vector = {}

    para i en imagenesSimples:
        etiqueta, nombre_etiqueta, predicciones = obtener_prediccion_RF(modeloRF, vector[i])
        predicciones_a_vector[predicciones[0][1]] = vector[i]

    # Encontrar las 5 predicciones más altas
    top_5_predicciones = ordenar(predicciones_a_vector.llaves(), reverso=Verdadero)[:5]

    para cada pred en top_5_predicciones:
        imprimir("Predicción:", pred)
        mostrar_imagen(vector_relacionado)
```

Figura 17. Pseudocódigo para obtener las imágenes simples con mayor certeza

Para cada uno de estos tres conjuntos de ejemplos explicativos, se llevarán a cabo pruebas con sujetos reales, y se evaluará la capacidad del modelo para generar explicaciones comprensibles y coherentes para las clasificaciones realizadas. Este enfoque diversificado de selección de ejemplos permite una evaluación completa de la capacidad del modelo para interactuar con usuarios reales y proporcionar explicaciones claras y útiles en diferentes contextos de uso.

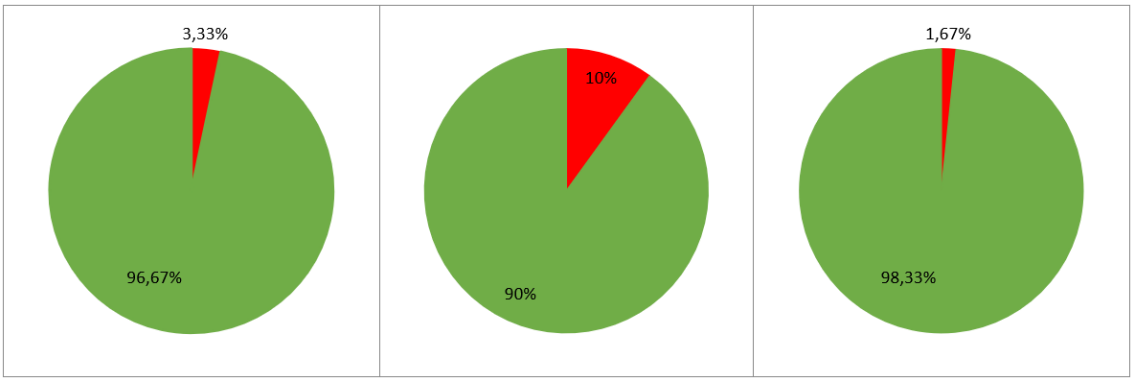
6.6 Análisis de resultados de pruebas con sujetos reales.

En esta sección, se presentarán los resultados obtenidos a través de las pruebas realizadas con sujetos, considerando las tres diferentes selecciones de imágenes utilizadas para recopilar datos. Estas selecciones se dividen en pares de categorías, donde a la izquierda se encuentran los resultados de las imágenes que representan ejemplos simples, en el centro se muestran los resultados de la selección aleatoria de imágenes, y a la derecha se exponen los resultados de las imágenes con mayor certeza.

Los resultados obtenidos en este estudio se recopilan a partir de una serie de pruebas llevadas a cabo con un grupo de poco más de 10 participantes de diversas edades y perfiles. En este conjunto de participantes formaban parte 2 individuos con más de 45 años, 2 jóvenes de 18 años y el resto de los participantes comprendían las edades de entre los 23 y 27 años. Entre los jóvenes, todos eran graduados universitarios, con la excepción de los dos de 18 años, que eran estudiantes de nuevo ingreso. En este grupo, se contó con la participación de 2 o 3 personas que tenían experiencia previa en el dibujo, mientras que otros participantes no tenían notables habilidades artísticas. La

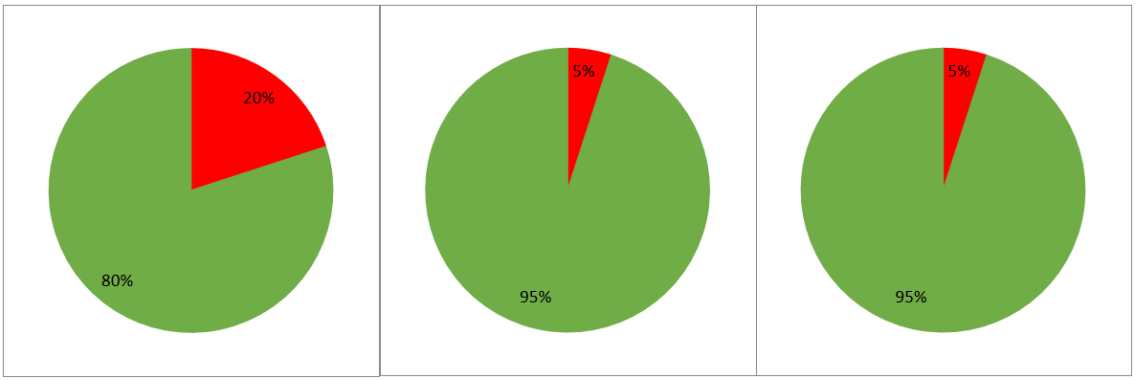
selección de esta diversidad de perfiles se hizo de manera deliberada, buscando personas con varios niveles de habilidad y pudiendo representar diferentes campos de especialización, dado que había personas graduadas en salud, educación, ingeniería y diseño.

El procedimiento de las pruebas consistió en mostrar imágenes a los participantes, divididas las imágenes entre parejas de categorias. Antes de mostrar las imágenes, se proporcionaba a los sujetos información sobre a qué par de categorías pertenecían las imágenes y se les otorgaba un máximo de 5 segundos para identificar la categoría correspondiente. En la mayoría de los casos, los participantes lograban identificar una de las dos categorías en un tiempo de 1 a 2 segundos, ya sea de manera correcta o incorrecta. Cada respuesta de los participantes fue registrada y documentada, permitiendo calcular el porcentaje de respuestas correctas en cada una de las selecciones de parejas para cada individuo.



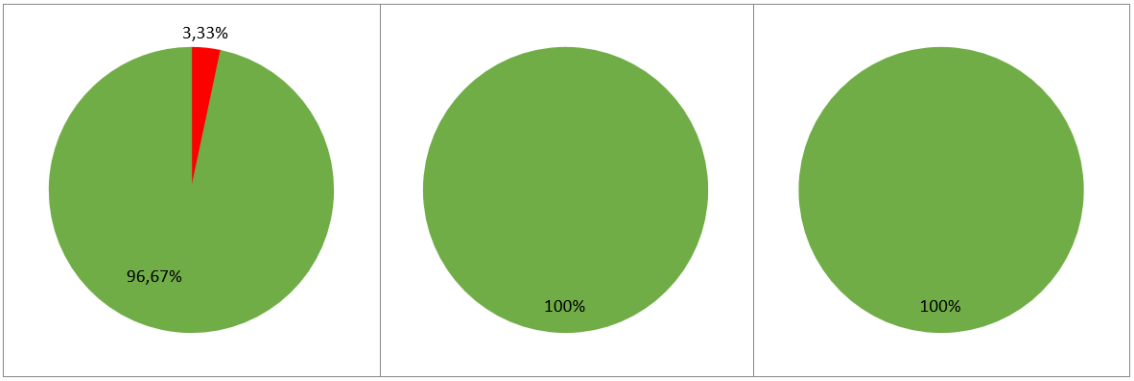
Gráfica 7. Porcentaje de acierto de la pareja 1

Para iniciar, examinemos los resultados obtenidos para la primera pareja de categorías: "coche" y "camión". Es notable que la mayoría de los sujetos lograron un reconocimiento bastante alto en esta prueba, a excepción de las imágenes seleccionadas de manera aleatoria, que presentan un ligero descenso en el porcentaje de acierto, aunque sigue siendo bastante elevado. Resulta crucial prestar especial atención al primer y último gráfico de esta sección, que indican el porcentaje de aciertos tanto para las imágenes más simples como para aquellas con mayor certeza dentro de las imágenes simples. Estos resultados sugieren que, en esta pareja en particular, la identificación de imágenes más simples con menos trazos resulta más sencilla para los participantes.



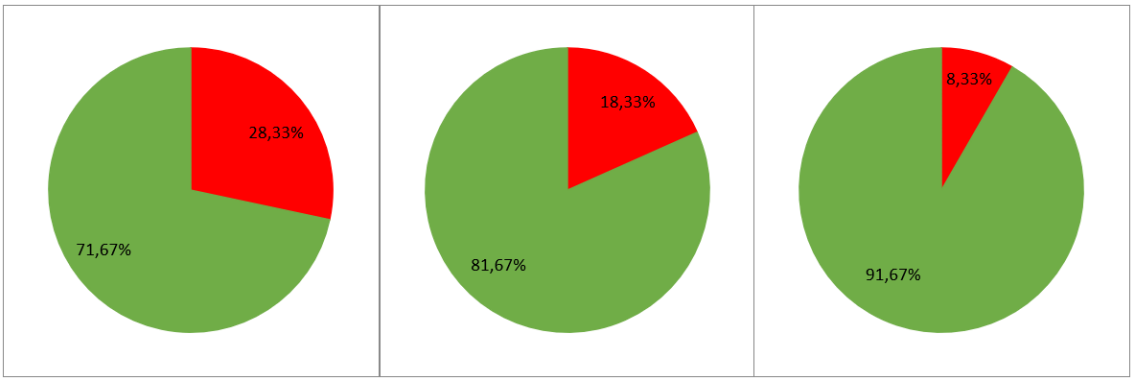
Gráfica 8. Porcentaje de acierto de la pareja 2

En el caso de la segunda pareja de categorías, "abeja" y "mariposa", se observa un rendimiento inferior en la identificación de imágenes más simples. Esta discrepancia podría deberse al hecho de que dada la similitud de estos insectos y su tamaño reducido, los usuarios tienden a crear dibujos de dimensiones más pequeñas, lo que complica su diferenciación, especialmente en imágenes simplificadas. Además, en esta pareja en particular, la distinción principal entre ambas categorías se relaciona principalmente con las alas de los insectos y posiblemente con los trazos del cuerpo de la abeja. Esto añade un nivel adicional de dificultad en el reconocimiento cuando las imágenes se vuelven más pequeñas.



Gráfica 9. Porcentaje de acierto de la pareja 3

En cuanto a la pareja "árbol" y "arbusto", se destaca un reconocimiento prácticamente perfecto por parte de los sujetos. Es evidente que al dibujar estas categorías existe una marcada diferencia entre ambas, especialmente en lo que respecta al tronco y la altura. Esta distinción clara facilita su reconocimiento por parte de los sujetos, ya que no se requieren muchos detalles y existe un factor diferenciador evidente.



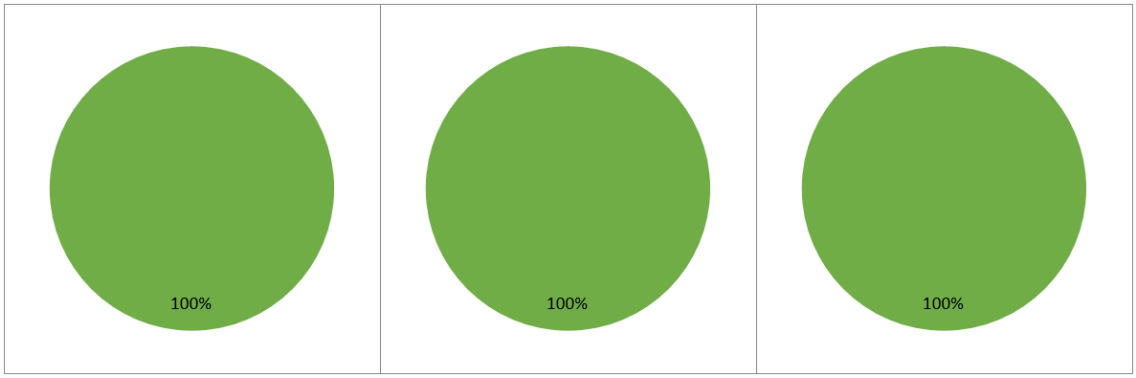
Gráfica 9. Porcentaje de acierto de la pareja 4

En este otro caso, la pareja de categorías "flor" y "árbol" se destaca como la más desafiante en términos de reconocimiento, ya que se observa un alto número de fallos en comparación con las demás parejas. Esto podría atribuirse a la alta similitud entre ambas categorías.

En el primer conjunto de imágenes, que consisten en las más simples, notamos que el reconocimiento es el más deficiente. Esta dificultad podría explicarse por la extrema similitud entre las categorías, lo que complica su reconocimiento cuando los dibujos son muy simples.

Sin embargo, cuando evaluamos el conjunto de imágenes obtenidas de forma aleatoria, observamos una mejora en el porcentaje de reconocimiento. Esto puede atribuirse a la inclusión de dibujos más complejos y con un mayor número de trazos, lo que proporciona más información distintiva.

Por último, al considerar el conjunto de imágenes con mayor certeza por parte del modelo, observamos los porcentajes más altos de reconocimiento. Esto podría sugerir que, incluso con imágenes simples, es posible lograr un alto nivel de reconocimiento, siempre y cuando la característica clave que diferencia ambas categorías esté bien representada en el dibujo.



Gráfica 10. Porcentaje de acierto de la pareja 5

En último lugar, examinamos la pareja de categorías "silla" y "mesa". En este caso, se observa un reconocimiento completo en todos los conjuntos de imágenes. Esto se debe a que ambas categorías no requieren muchos detalles para diferenciarse, y existe una clara distinción entre ambas clases debido a su simplicidad. En esta pareja en particular, los usuarios tienden a dibujar de la manera más sencilla posible, ya que la diferencia entre ambas categorías es bastante evidente.

7. Discusión de Resultados

7.1 Interpretación de los Resultados

En la interpretación de los resultados, varios aspectos destacan en cuanto a la capacidad de reconocimiento de categorías por parte de los modelos de CNN y los sujetos humanos. En primer lugar, se observa que, cuando las categorías presentan diferencias notables y características distintivas, mantener los dibujos simples se revela como una estrategia efectiva. Esto se debe a que en situaciones donde las diferencias son evidentes, un enfoque simplificado facilita la identificación de las características clave que definen cada categoría.

Por otro lado, en los casos de categorías extremadamente similares, la simplicidad puede llevar a confusión, a menos que existan características distintivas claramente reconocibles. Por ejemplo, en el caso de los gatos y los perros, las orejas pueden ser una característica distintiva que simplifica su identificación. Sin embargo, en ausencia de rasgos diferenciadores, los dibujos más complejos, con más detalles, pueden ser necesarios para evitar confusiones en la identificación.

Por consecuencia, un punto importante para considerar es que la elección de mantener un dibujo simple o complejo depende en gran medida de si las categorías comparables tienen características que resalten una sobre la otra. Si existe una característica destacada, simplificar el dibujo para enfocarse en esa característica particular puede ser una estrategia efectiva para facilitar la identificación.

Asimismo, se destaca la relevancia de la selección cuidadosa de las categorías de imágenes al implementar un modelo de CNN para tareas de clasificación. La elección adecuada de categorías puede influir en la efectividad del modelo y en la facilidad con la que las personas pueden reconocerlas, lo que subraya la importancia de esta decisión en el proceso de diseño.

Los resultados también revelan diferencias interesantes entre el rendimiento de los modelos de CNN y los sujetos humanos. Por ejemplo, en el caso "P5", se observa que el rendimiento humano supera al del modelo, mientras que en "P4",

el modelo supera a los humanos. Estas discrepancias sugieren que los modelos de CNN y los seres humanos pueden enfocarse en diferentes aspectos al reconocer categorías, lo que agrega complejidad al proceso de reconocimiento.

Además, los resultados también indican cómo el reconocimiento varía según la similitud de los dibujos presentes en las parejas de categorías. Categorías con dibujos altamente similares pueden resultar más desafiantes tanto para los modelos como para los seres humanos, lo que resalta la influencia de la similitud en la tarea de reconocimiento.

Por último, se destaca que las habilidades individuales de dibujo también pueden influir en el reconocimiento. Algunos sujetos pueden tener una mayor facilidad para representar ciertas categorías con mayor precisión, lo que agrega una dimensión adicional de variabilidad en los resultados de las pruebas. Sin embargo, este punto no afectaría al entrenamiento ni reconocimiento mediante modelos.

7.2 Limitaciones del Enfoque Propuesto

A pesar de los avances y resultados prometedores obtenidos en este estudio sobre la generación de explicaciones en modelos de predicción de objetos en garabatos, es importante destacar algunas limitaciones clave que deben tenerse en cuenta:

- **Dependencia de la Calidad de los Datos de Entrada**: La calidad de las explicaciones generadas por los modelos de CNN y RF depende en gran medida de la calidad de los datos de entrada, es decir, de la precisión con la que se reconocen los garabatos. Si las imágenes de entrada contienen ruido o ambigüedad, las explicaciones pueden resultar menos claras o precisas.
- **Dependencia del Tamaño del Conjunto de Datos**: La capacidad de generalización y la calidad de las explicaciones generadas pueden verse afectadas por el tamaño del conjunto de datos utilizado para entrenar los modelos. En este estudio, se ha utilizado un conjunto de datos de tamaño moderado. Ampliar el conjunto de datos podría mejorar aún más la eficacia del modelo.

- **Sujetos Reales y Evaluación Humana**: Las pruebas con sujetos reales para evaluar la calidad de las explicaciones son esenciales pero pueden introducir cierto grado de subjetividad. Las diferencias individuales en la comprensión de las explicaciones pueden influir en los resultados.
- **Complejidad de los Dibujos**: El enfoque se ha centrado principalmente en dibujos relativamente simples y directos. Dibujos más complejos o abstractos pueden presentar desafíos adicionales en términos de reconocimiento y generación de explicaciones.
- **Generalización a Otros Dominios**: Si bien el enfoque se ha aplicado a la predicción de objetos en dibujos, su extensión a otros dominios puede requerir ajustes significativos y no estar garantizada su eficacia.
- **Efectos de la Escala y Resolución**: El modelo de CNN utilizado podría verse limitado al reconocer imágenes a diferentes escalas y resoluciones. Podría tener dificultades con dibujos muy pequeños o con detalles finos, lo que afectaría la precisión de las explicaciones.
- **Sesgo en los Datos de Entrenamiento**: Si el conjunto de datos de entrenamiento contiene sesgos inherentes, como una representación desigual de ciertas categorías de garabatos, esto podría afectar la calidad de las explicaciones y la capacidad del modelo para generalizar de manera justa.
- **Generalización a Idiomas y Culturas Diversas**: La generalización de los modelos a garabatos creados por personas de diferentes idiomas y culturas podría ser un desafío, ya que el significado y la representación visual pueden variar ampliamente.

En resumen, a pesar de los logros alcanzados en la generación de explicaciones en modelos de predicción de objetos en dibujos, estas limitaciones subrayan la necesidad continua de investigación y desarrollo en este campo. Abordar estas limitaciones puede conducir a mejoras adicionales en la comprensión y la utilidad de los modelos de aprendizaje automático en aplicaciones de interpretación de garabatos y otros campos relacionados.

8. Conclusiones

8.1 Resumen de Resultados

En un mundo que avanza hacia una creciente dependencia de la tecnología y la inteligencia artificial, este estudio subraya la interconexión entre la creatividad humana y la capacidad de las máquinas para aprender. Si bien hemos demostrado cómo los modelos de redes neuronales y Random Forest pueden clasificar dibujos de diversas categorías, es importante recordar que detrás de cada trazo o línea está la imaginación humana.

El estudio arroja luz sobre el impacto de la simplicidad en la representación visual de objetos. Hemos observado que el uso de imágenes simples puede mejorar el entendimiento y el rendimiento de los modelos, aunque este beneficio puede variar significativamente según las categorías específicas consideradas. Se concluye que la elección de categorías a comparar desempeña un papel crítico en la eficacia de este enfoque.

En lo que respecta a los modelos, hemos constatado que las imágenes simples son efectivas para entrenarlos y lograr el reconocimiento de categorías, al menos en el contexto de comparaciones entre dos tipos de categorías, como lo demostramos en nuestras pruebas de parejas. No obstante, en el caso de los seres humanos, algunas categorías todavía requieren imágenes más complejas y detalladas para permitir una distinción clara.

También se llega a la conclusión de que los humanos tienden a considerar ciertas características distintivas de los objetos en la vida real e intentan relacionar dichas características a la hora de reconocer una imagen, por ello a la hora de reconocer imágenes, se requieren detalles más complejos dependiendo la categoría. Estas características no siempre están presentes en los bocetos, pero los modelos no tienen ese inconveniente a la hora de reconocer, dado que se basan principalmente en encontrar patrones en los dibujos.

Estos resultados subrayan la necesidad de una cuidadosa elección de categorías al implementar modelos de inteligencia artificial para tareas de clasificación visual. También destacan las áreas donde los modelos podrían

beneficiarse de mejoras, como la incorporación de características distintivas en el proceso de entrenamiento, lo que podría acercar aún más la brecha entre la visión de las máquinas y la percepción humana de las imágenes.

8.2 Contribuciones del TFM

Este trabajo de investigación ha profundizado en la influencia de las imágenes simples en la representación visual de objetos y su impacto en el desempeño de modelos de aprendizaje automático. Esta exploración ha proporcionado una nueva perspectiva sobre cómo la simplicidad puede ser una variable relevante en la clasificación de objetos.

El estudio también ha permitido un análisis en profundidad de cómo los seres humanos y los modelos de inteligencia artificial abordan la tarea de identificar objetos. Estas comparaciones revelan diferencias fundamentales en los procesos cognitivos y de aprendizaje, mostrando las fortalezas y limitaciones inherentes a ambos enfoques. Esto resulta especialmente valioso para comprender cómo las capacidades de las máquinas pueden complementar y mejorar la percepción humana.

Además, se han llevado a cabo pruebas específicas utilizando parejas de categorías de objetos similares. Este enfoque de evaluación proporciona información detallada sobre la capacidad de los modelos para discriminar entre objetos que comparten similitudes visuales. Estas pruebas también ayudan a identificar patrones en la capacidad de reconocimiento de las máquinas y proporcionaron ideas valiosas sobre cómo el contexto de las categorías puede influir en el rendimiento de los modelos.

En resumen, este TFM ha contribuido a una comprensión más profunda de cómo la simplicidad, la interacción entre modelos de máquina y la elección de categorías influyen en el reconocimiento de objetos.

8.3 Relación del trabajo desarrollado con los estudios cursados

La relación del trabajo desarrollado en este TFG con los estudios cursados es una consideración fundamental, por lo que este análisis permite demostrar la

capacidad para aplicar los conocimientos adquiridos a lo largo de sus estudios para abordar problemas reales en el mundo laboral.

El enfoque de este proyecto se basa en la aplicación de conceptos y técnicas de inteligencia artificial y aprendizaje automático, específicamente en el campo de la explicabilidad de la inteligencia artificial (XAI). A lo largo de mis estudios, he adquirido ciertos conocimientos sobre estas áreas, que incluyen la comprensión de algoritmos de aprendizaje automático y el procesamiento de datos, aunque en el campo del aprendizaje automático sí que han sido un poco más escasos. Estos conocimientos previos me han proporcionado una base para abordar el problema de la explicabilidad en la inteligencia artificial y su relación con la simplicidad en los dibujos.

Por otro lado, de este proyecto también se beneficia de mi formación en programación y desarrollo de software, lo que me ha permitido implementar cambios eficazmente sobre los modelos de aprendizaje automático y realizar las funciones necesarias para el análisis de datos y para la preparación de los datos. Además, mi formación en la recopilación y organización de datos ha sido útil para la realización de las pruebas con los participantes y la recopilación de datos.

En cuanto a las competencias transversales, la elaboración de este TFG ha requerido habilidades de investigación, análisis crítico y resolución de problemas. La capacidad para diseñar y llevar a cabo pruebas con participantes, así como para analizar y comunicar los resultados, también ha sido fundamental.

En resumen, este proyecto demuestra la aplicación de conocimientos técnicos adquiridos durante mis estudios en áreas como inteligencia artificial, aprendizaje automático y programación. Además, ha requerido el desarrollo de competencias transversales como la investigación, el análisis crítico y la resolución de problemas, que son esenciales en el mundo laboral actual.

8.4 Trabajo Futuro

En vista de los resultados obtenidos y las oportunidades identificadas durante este estudio, se pueden sugerir varias direcciones para futuras investigaciones. Una de las áreas de mejora más notables radica en la ampliación de las categorías de imágenes en el conjunto de datos. Este enfoque permitiría evaluar

aún más la capacidad de los modelos para reconocer una variedad más amplia de objetos o conceptos, lo que podría ser especialmente relevante en aplicaciones del mundo real. La inclusión de categorías adicionales podría proporcionar información valiosa sobre cómo se actuarían los modelos en situaciones donde se necesite un reconocimiento más amplio.

Continuar explorando la optimización de hiperparámetros en la arquitectura de los modelos de aprendizaje automático representa otra vía de mejora. Esto podría implicar ajustar la arquitectura de la CNN o afinar los parámetros del modelo de RF para lograr un rendimiento aún más preciso y eficiente. La optimización de hiperparámetros ofrece la posibilidad de perfeccionar la capacidad de los modelos para diferenciar categorías de manera más efectiva y eficiente.

Además de las métricas de evaluación utilizadas en este estudio, se podría considerar la inclusión de otras métricas adicionales para obtener una comprensión más completa del rendimiento del modelo. Métricas como la matriz de confusión, el área bajo la curva ROC (AUC-ROC) o medidas de sensibilidad podrían proporcionar una visión detallada de cómo los modelos están llevando a cabo la clasificación y destacar áreas específicas que requieren atención adicional.

Finalmente, para comprender mejor la interacción entre las personas y los sistemas de reconocimiento de dibujos, se podrían llevar a cabo estudios de usuario más exhaustivos. Estos estudios podrían investigar cómo personas de diferentes edades, niveles de habilidad o familiaridad con las categorías interactúan con estos sistemas. Esta información sería crucial para adaptar los sistemas de reconocimiento y satisfacer las necesidades y expectativas de una variedad de usuarios y contextos de uso.

9. Anexo

9.1 Detalles de Implementación

En el apartado del anexo, vamos a adentrarnos en el corazón del proyecto, donde se va a indicar el pseudocódigo del código utilizado en el desarrollo. Mediante el pseudocódigo se va a ofrecer una visión más detallada de cómo se estructura y ejecuta el programa utilizado tanto en la creación de los modelos como las pruebas de estos.

```
Procedimiento vector_a_raster(vector_imagenes, lado=28, diametro_linea=16, relleno=16,
                              color_fondo=(0,0,0), color_trazo=(1,1,1)):

    lado_original = 256.

    superficie = CrearSuperficieImagen(Formato=ARGB32, Ancho=lado, Alto=lado)
    contexto = CrearContexto(superficie)
    ConfigurarSuavizado(contexto, MejorSuavizado)
    ConfigurarPuntaLinea(contexto, PuntaRedonda)
    ConfigurarUnionLinea(contexto, UnionRedonda)
    ConfigurarAnchoLinea(contexto, diametro_linea)

    # Escalar para que coincida con el nuevo tamaño
    # Agregar relleno en los bordes para el diámetro de la línea
    # y agregar relleno adicional para la suavización de bordes
    relleno_total = relleno * 2. + diametro_linea
    nueva_escala = lado / (lado_original + relleno_total)
    EscalarContexto(contexto, nueva_escala, nueva_escala)
    TraducirContexto(contexto, relleno_total / 2., relleno_total / 2.)

    imagenes_raster = []
    Para cada imagen_vector en vector_imagenes:
        # Limpiar el fondo
        ConfigurarFuenteRGB(contexto, color_fondo)
        Pintar(contexto)

        caja = Maximo(vector_imagen).en_el_eje(1)
        desplazamiento = ((lado_original, lado_original) - caja) / 2.
        desplazamiento = Redefinir(desplazamiento, -1, 1)
        centrado = [trazo + desplazamiento para trazo en imagen_vector]

        # Dibujar trazos, esta es la parte más intensiva en CPU
        ConfigurarFuenteRGB(contexto, color_trazo)
        Para cada x_vector, y_vector en centrado:
            MoverA(contexto, x_vector[0], y_vector[0])
            Para cada x, y en zip(x_vector, y_vector):
                LineaA(contexto, x, y)
            Trazo(contexto)

        datos = ObtenerDatos(superficie)
        imagen_raster = Copiar(ComoArray(datos)[::4])
        imagenes_raster.Agregar(imagen_raster)

    Devolver ComoArray(imagenes_raster).ComoTipo('float32') / 255.0


Función cargar_datos():

    # Diccionario para URL y etiquetas de clase
    diccionario_clases = {}

    lista_dibujos = []
    para cada dibujo en obtener_dibujos('google_data/full_binary_car.bin'):
        # Agregar el dibujo a la lista
        lista_dibujos.agregar(dibujo['image'])
```

```
    imagenes_raster = vector_a_raster(lista_dibujos)
    diccionario_clases['coche'] = imagenes_raster

    lista_dibujos2 = []
    para cada dibujo en obtener_dibujos('google_data/full_binary_truck.bin'):
        # Agregar el dibujo a la lista
        lista_dibujos2.agregar(dibujo['image'])

    imagenes_raster2 = vector_a_raster(lista_dibujos2)
    diccionario_clases['camión'] = imagenes_raster2

    # Generar etiquetas y agregar etiquetas a los datos cargados
    para i, (clave, valor) en enumerar(diccionario_clases.elementos()):
        valor = valor.convertir_a('float32') / 255.0
        si i == 0 entonces
            diccionario_clases[clave] = concatenar_columnas(valor, ceros(len(valor)))
        sino
            diccionario_clases[clave] = concatenar_columnas(valor, i * unos(len(valor)))

    lista_datos = []
    para cada clave, valor en diccionario_clases.elementos():
        lista_datos.agregar(valor)

    doodles = concatenar(lista_datos)

    # Dividir los datos en características y etiquetas de clase (X e y respectivamente)
    y = doodles[:, -1].convertir_a('float32')
    X = doodles[:, :784]

    # Dividir cada conjunto de datos en conjuntos de entrenamiento/prueba
    X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = dividir_conjuntos(X, y,
        tamaño_prueba=0.3, semilla_aleatoria=1)

    Devolver X_entrenamiento, y_entrenamiento, X_prueba, y_prueba


Procedimiento crearModeloCNN(X_train, y_train, X_test, y_test):
    # Parsear argumentos de línea de comandos
    learning_rate = 0.001
    epochs = 30
    weight_decay = 0
    dropout = 0.0
    mini_batches = 1000
    optimizer = 'SGD'
    gpu = Falso

    learning_rate = ObtenerArgumentoFlotante('--learning_rate', 0.001)
    epochs = ObtenerArgumentoEntero('--epochs', 30)
    weight_decay = ObtenerArgumentoFlotante('--weight_decay', 0)
    dropout = ObtenerArgumentoFlotante('--dropout', 0.0)
    mini_batches = ObtenerArgumentoEntero('--mini_batches', 1000)
    optimizer = ObtenerArgumento('--optimizer', 'SGD', opciones=['SGD', 'Adam'])
    gpu = ObtenerArgumentoBooleano('--gpu')

    resultado = ParsearArgumentos()
```

```
    arquitectura = resultado.arquitectura
    n_chunks = 824

    si gpu == Verdadero entonces
        dispositivo = 'cuda'
    sino
        dispositivo = 'cpu'

    si resultado.save_dir == ' ' entonces
        ruta_guardado = 'checkpoint.pth'
    sino
        ruta_guardado = resultado.save_dir + '/' + 'checkpoint.pth'

    # Agregar imágenes reflejadas y rotadas al conjunto de datos
    si resultado.add_data == Verdadero entonces
        X_train, y_train = agregar_imagenes_reflejadas_y_rotadas(X_train, y_train)

    # Guardar conjuntos de datos en disco si es necesario
    guardar_datos(X_train, y_train, X_test, y_test, forzar = resultado.add_data)

    # Convertir a tensores
    entrenamiento = ConvertirATensor(X_train).Flotante()
    etiquetas = ConvertirATensor(y_train).Entero()
    prueba = ConvertirATensor(X_test).Flotante()
    etiquetas_prueba = ConvertirATensor(y_test).Entero()

    # Hiperparámetros para nuestra red
    tamaño_entrada = 784
    tamaños_ocultos = [128, 100, 64]
    tamaño_salida = 2

    # Construir modelo
    modelo = construir_modelo(tamaño_entrada, tamaño_salida, tamaños_ocultos,
            arquitectura = arquitectura, dropout = dropout)

    # Ajustar modelo
    ajustar_modelo(modelo, entrenamiento, etiquetas, epochs = epochs, n_chunks = n_chunks,
                learning_rate = learning_rate, weight_decay = weight_decay,
                optimizer = optimizer)
    Devolver modelo


Procedimiento ajustarModelo(modelo, X_entrenamiento, y_entrenamiento, epochs = 30, n_chunks = 824,
                            learning_rate = 0.001, weight_decay = 0, optimizador = 'SGD'):
    criterio = CrearCriterioEntropiaCruzada()

    si optimizador == 'SGD' entonces
        optimizador = CrearOptimizadorSGD(modelo.parametros(), tasa_aprendizaje=learning_rate,
                decaimiento_peso=weight_decay)
    sino
        optimizador = CrearOptimizadorAdam(modelo.parametros(), tasa_aprendizaje=learning_rate,
                decaimiento_peso=weight_decay)

    imprimir_cada = 100

    pasos = 0
```

```
    para cada epoca en el_rango(epochs):
        perdida_corriente = 0

        X_entrenamiento, y_entrenamiento = barajar(X_entrenamiento, y_entrenamiento)

        imagenes = dividir_en_trozos(X_entrenamiento, n_chunks)
        etiquetas = dividir_en_trozos(y_entrenamiento, n_chunks)

        para cada i en el_rango(n_chunks):
            pasos += 1

            reiniciar_optimizador(optimizador)

            # Pases hacia adelante y hacia atrás
            salida = modelo.pase_hacia_adelante(imagenes[i])
            perdida = calcular_perdida(salida, etiquetas[i].aplastar())
            propagar_perdida_hacia_atras(perdida)
            aplicar_optimizador(optimizador)

            perdida_corriente += obtener_valor(perdida)

            si pasos % imprimir_cada == 0 entonces
                imprimir("Época: {}/{}... ".format(epoca+1, epochs),
                      "Perdida: {:.4f}".format(perdida_corriente/imprimir_cada))

                perdida_corriente = 0
```

```
Procedimiento probarModeloCNN(modeloCNN):
    # Poner el modelo en modo de evaluación
    modeloCNN.eval()

    dibujos_lista_coche = []
    dibujos_lista_camion = []

    # Ordenar por simplicidad
    desempacar_coche = ordenar(desempacar_dibujos('google_data/full_binary_car.bin'),
                               clave=lambda x: x['n_strokes'])
    para cada dibujo en desempacar_coche:
        # Agregar el dibujo a la lista
        agregarAlista(dibujos_lista_coche, dibujo['image'])

    # Ordenar por simplicidad
    desempacar_camion = ordenar(desempacar_dibujos('google_data/full_binary_truck.bin'),
                               clave=lambda x: x['n_strokes'])
    para cada dibujo en desempacar_camion:
        # Agregar el dibujo a la lista
        agregarAlista(dibujos_lista_camion, dibujo['image'])

    raster_imagenes_coche = vector_a_raster(dibujos_lista_coche)
    raster_imagenes_camion = vector_a_raster(dibujos_lista_camion)
    vector_coche = []
    vector_camion = []
```

```
    para cada fila en raster_imagenes_coche:
        etiqueta, nombre_etiqueta, predicciones = obtener_prediccion_CNN(modeloCNN, fila)
        si etiqueta == 0 entonces
            agregarAlista(vector_coche, fila)

    para cada fila en raster_imagenes_camion:
        etiqueta, nombre_etiqueta, predicciones = obtener_prediccion_CNN(modeloCNN, fila)
        si etiqueta == 1 entonces
            agregarAlista(vector_camion, fila)

    Devolver vector_coche, vector_camion


Procedimiento crearRandomForestClassifier(X_train, y_train):
    # Instanciar el clasificador de bosque aleatorio
    modelo = ClasificadorBosqueAleatorio(n_estimators=100, max_depth=None)

    # Ajustar el clasificador
    AjustarModelo(modelo, X_train, y_train)

    Devolver modelo


Procedimiento probarRandomForestClassifier(modelo, X_entrenamiento, y_entrenamiento, X_prueba,
                                           y_prueba):

    # Obtener predicciones para el conjunto de entrenamiento y prueba
    y_predicciones_entrenamiento = predecir(modelo, X_entrenamiento)
    y_predicciones_prueba = predecir(modelo, X_prueba)

    # Calcular el puntaje de precisión para el conjunto de entrenamiento y prueba
    precisión_entrenamiento = calcular_puntaje_precisión(y_entrenamiento,
                                                         y_predicciones_entrenamiento)
    precisión_prueba = calcular_puntaje_precisión(y_prueba, y_predicciones_prueba)

    Devolver precisión_entrenamiento, precisión_prueba


Procedimiento pruebasModeloRF(modelRF, vector):
    pred_a_vector = {}

    # Encontrar los 10 valores de predicción más simples
    para i en el_rango(10):
        mostrar_imagen(vector[i])

    # Seleccionar 10 valores de predicción aleatorios
    seleccionar_valores_aleatorios(vector, 10)

    para cada i en el_rango(longitud(vector)):
        etiqueta, nombre_etiqueta, predicciones = obtener_prediccion_RF(modelRF, vector[i])

        pred_a_vector[predicciones[0][0]] = vector[i]

    # Encontrar los 5 valores de predicción más altos
    top_10_preds = ordenar(pred_a_vector.llaves(), reverso=Verdadero)[:5]
```

```
    para cada pred en top_10_preds:
        vector_relacionado = pred_a_vector[pred]
        imprimir("Predicción:", pred)
        mostrar_imagen(vector_relacionado)

    para cada pred en bot_10_preds:
        vector_relacionado = pred_a_vector[pred]
        imprimir("Predicción:", pred)
        mostrar_imagen(vector_relacionado)


Procedimiento main():
    # Crear el modelo CNN
    X_entrenamiento, y_entrenamiento, X_prueba, y_prueba = cargar_datos()
    modeloCNN = crearModeloCNN(X_entrenamiento, y_entrenamiento, X_prueba, y_prueba)

    # Probar el modelo CNN
    vector_coche, vector_camion = probarModeloCNN(modeloCNN)

    vector = [10, 50, 100, 200, 500, 1000, 5000, 10000, 15000, 20000, 25000, 30000]

    para cada elemento en vector:
        X_entrenamiento_RF, y_entrenamiento_RF, X_prueba_RF, y_prueba_RF = cargar_datos_RF_prueba(
            np.array(vector_coche[:elemento]), np.array(vector_camion[:elemento]))

        # Crear el modelo RF
        modeloRF = crearRandomForestClassifier(X_entrenamiento_RF, y_entrenamiento_RF)

        probarRandomForestClassifier(modeloRF)

    # Probar el modelo RF con vector_1
    pruebasModeloRF(modeloRF, vector_1)

    # Probar el modelo RF con vector_2
    pruebasModeloRF2(modeloRF, vector_2)
```

El pseudocódigo mostrado forma parte de la parte del corazón del proyecto, tal y como se ha comentado. Sin embargo, es importante señalar que existen otras funciones y componentes implementados en el proyecto que no se reflejan en este pseudocódigo. Aunque estas funciones no son esenciales para el funcionamiento fundamental del proyecto

Un ejemplo destacado de estas funciones adicionales se relaciona con el almacenamiento y la carga de datos. Estas funciones fueron desarrolladas para optimizar la eficiencia y reducir tanto los costos como los tiempos asociados con la obtención de datos desde la plataforma QuickDraw. Además, también se han implementado funciones para el almacenamiento y la carga de los modelos, lo que garantiza que los modelos entrenados se puedan conservar y utilizar en futuras instancias del proyecto, evitando también la necesidad de volver a entrenarlos desde cero. Sin embargo, también hay funciones ajenas al almacenamiento y la carga de datos, un ejemplo de ello sería la función para mostrar imágenes que se utilizaron en el proceso de entrenamiento de los modelos.

9.2 Objetivos de Desarrollo Sostenible de la Agenda 2023

A continuación se va a mostrar el grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza**				X
ODS 2. **Hambre cero**				X
ODS 3. **Salud y bienestar**				
ODS 4. **Educación de calidad**		X		
ODS 5. **Igualdad de género**				X
ODS 6. **Agua limpia y saneamiento**				X
ODS 7. **Energía asequible y no contaminante**				X
ODS 8. **Trabajo decente y crecimiento económico**			X	
ODS 9. **Industria, innovación e infraestructuras**	X			
ODS 10. **Reducción de las desigualdades**				X
ODS 11. **Ciudades y comunidades sostenibles**			X	
ODS 12. **Producción y consumo responsables**				X
ODS 13. **Acción por el clima**				X
ODS 14. **Vida submarina**				X
ODS 15. **Vida de ecosistemas terrestres**				X
ODS 16. **Paz, justicia e instituciones sólidas**				X
ODS 17. **Alianzas para lograr objetivos**				X

Descripción de la alineación del TFM con los ODS con un grado de relación más alta.

El proyecto se alinea principalmente con dos de los Objetivos de Desarrollo Sostenible (ODS): el ODS 4, que busca "Educación de Calidad", y el ODS 9, que se enfoca en "Industria, Innovación e Infraestructuras". A continuación, se detalla cómo este proyecto se relaciona con estos ODS:

El proyecto contribuye directamente al ODS 4, que busca garantizar una educación de calidad para todos. A través del desarrollo de herramientas y modelos de aprendizaje automático este proyecto puede tener un impacto

significativo en la educación al ayudar a automatizar tareas de evaluación y proporcionar retroalimentación oportuna y precisa a los estudiantes.

Además, al utilizar datos recopilados de QuickDraw, el proyecto promueve el acceso a recursos educativos en línea, lo que está en línea con la idea de utilizar la tecnología con fines educativos y hacerla accesible para un público más amplio.

Por otra parte, el ODS 9 se enfoca en promover la construcción de infraestructuras y el fomento de la innovación para el desarrollo sostenible. Este proyecto se alinea con este objetivo dado que se desarrollan modelos de aprendizaje automático que pueden utilizarse en una variedad de aplicaciones industriales y educativas. La innovación aquí radica en la aplicación de técnicas de inteligencia artificial para la clasificación de imágenes, lo que puede impulsar la eficiencia en diversas industrias.

Además, la infraestructura necesaria para ejecutar estos modelos también contribuye al ODS 9, ya que requiere recursos tecnológicos y digitales para su implementación.

10. Bibliografía

[1] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On Pixel-Wise explanations for Non-Linear Classifier decisions by Layer-Wise relevance propagation. *PLOS ONE*, *10*(7), e0130140. https://doi.org/10.1371/journal.pone.0130140

[2] Calle, M. (2021). Métodos de explicabilidad de la inteligencia artificial en salud. *Instituto de Ingeniería del Conocimiento*. https://www.iic.uam.es/lasalud/metodos-explicabilidad-inteligencia-artificial-en-salud/

[3] *Explainable Artificial Intelligence (XAI): How to make image analysis deep learning models transparent*. (2022, 27 noviembre). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10003813

[4] Googlecreativelab. (s. f.). *GitHub - GoogleCreativeLab/QuickDraw-Dataset: Documentation on how to access and use the Quick, Draw! dataset.* GitHub. https://github.com/googlecreativelab/quickdraw-dataset

[5] *Introducing a web component and data API for quick, draw!* (s. f.). Google Open Source Blog. https://opensource.googleblog.com/2018/11/introducing-web-component-and-data-api-for-quick-draw.html

[6] Keisukeirie. (s. f.). *GitHub - keisukeirie/quickdraw_prediction_model: This is my repository for the Quick Draw Prediction Model project*. GitHub. https://github.com/keisukeirie/quickdraw_prediction_model

[7] Lexie88rus. (s. f.). *GitHub - lexie88rus/Quick-draw-image-recognition: Recognition of quick draw doodles*. GitHub. https://github.com/Lexie88rus/quick-draw-image-recognition

[8] Luisquintanilla. (2023, 10 mayo). *Introducción al aprendizaje profundo - ML.NET*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/machine-learning/deep-learning-overview#deep-learning-vs-machine-learning

[9] Lundberg, S. (2017, 22 mayo). *A unified approach to interpreting model predictions*. arXiv.org. https://arxiv.org/abs/1705.07874

[10] *On the use of XAI for CNN Model Interpretation: a remote sensing case study*. (2022, 18 diciembre). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10089337

[11] *¿Qué es el aprendizaje automático?* | *Google Cloud* | *Google Cloud*. (s. f.). Google Cloud. https://cloud.google.com/learn/what-is-machine-learning?hl=es-419

[12] *Random Forest Python*. (s. f.). https://cienciadedatos.net/documentos/py08_random_forest_python.html

[13] *Sklearn.ensemble.RandomForestClassifier*. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[14] *Sklearn.metrics.accuracy_score*. (s. f.). scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[15] Team, D. (2022a, agosto 1). *Random Forest: Bosque aleatorio. Definición y funcionamiento*. Formation Data Science | DataScientest.com. https://datascientest.com/es/random-forest-bosque-aleatorio-definicion-y-funcionamiento

[16] Team, D. (2022b, septiembre 20). *Convolutional Neural Network : Definición y funcionamiento*. Formation Data Science | DataScientest.com. https://datascientest.com/es/convolutional-neural-network-es

[17] Vallandingham, J. (s. f.). *How long does it take to (Quick) draw a dog?* http://vallandingham.me/quickdraw/

[18] Zeiler, M. D. (2013, 12 noviembre). *Visualizing and understanding convolutional networks*. arXiv.org. https://arxiv.org/abs/1311.2901

[19] Sundararajan, M. (2017, 4 marzo). *Axiomatic attribution for deep networks*. arXiv.org. https://arxiv.org/abs/1703.01365

[20] *Overview — Pycairo Documentation*. (s. f.). https://pycairo.readthedocs.io/en/latest/

[21] M. S. Alfarzaeai, Q. Niu, J. Zhao, R. M. A. Eshaq and E. Hu, "Coal/Gangue Recognition Using Convolutional Neural Networks and Thermal Images," in IEEE Access, vol. 8, pp. 76780-76789, 2020, doi: 10.1109/ACCESS.2020.2990200.

[22] datos.gob.es. (2023, 5 abril). Iniciativas para entrenar modelos de machine learning con datos abiertos. *datos.gob.es*. https://datos.gob.es/es/blog/iniciativas-para-entrenar-modelos-de-machine-learning-con-datos-abiertos