Transferencia de aprendizaje con VGG16 para la evaluación del algoritmo de explicación de IA LIME

Tesis presentada para obtener el título de:

Licenciatura en Ingeniería Mecatrónica

Presenta:

Rafael Solar Hernández

Asesores:

Director: Dr. Enrique Rafael García Sánchez (BUAP)

Codirectora: Dra. María del Pilar Gómez Gil (INAOE)

Benemérita Universidad Autónoma de Puebla

Agosto, 2023

Puebla, Puebla

Abstract

Artificial vision applications exemplify the remarkable achievements of deep learning methods, which belong to the field of Artificial Intelligence (AI). In particular, Deep Convolutional Neural Networks (DCNN) have shown remarkable performance in image classification and artificial vision. However, the clear understanding of these models by their users and designers is still one of the main barriers to the practical implementation of AI. One tool for obtaining some explainability of a DCNN model is LIME (Local Interpretable Model-Agnostic Explanations), which basically consists of disturbing the input of a classification system to observe how its prediction changes. Although LIME has demonstrated its efficacy in specific scenarios, it is crucial to conduct a more thorough evaluation of its actual effectiveness to improve user confidence in artificial vision systems that DCNN.

In order to contribute to this field of study, this research project carries out a detailed analysis of the behavior of the LIME algorithm in a simple image classification task based on natural images. In addition, a web interface was developed that shows the classifier and LIME results, given an image, which allows users to analyze different types of images and to observe the congruence of the explanation of LIME with respect to the way in which humans would explain that classification.

Resumen

Las aplicaciones de visión artificial son uno de los ejemplos mas contundentes del éxito de las técnicas de aprendizaje profundo, las cuales pertenecen al campo de la Inteligencia Artificial (IA). En particular, las Redes Neuronales Convolucionales Profundas (DCNN por sus siglas en inglés) han demostrado un notable desempeño en la clasificación de imágenes y visión artificial. Sin embargo, en la actualidad el claro entendimiento de estos modelos por parte de sus usuarios y diseñadores sigue siendo una de las principales barreras para la implementación práctica de la IA. Una herramienta para obtener cierta explicabilidad de un modelo DCNN es LIME (Explicaciones Independientes de Modelo Interpretables de Forma Local, por sus siglas del inglés), el cual básicamente consiste en perturbar la entrada de un sistema de clasificación para observar cómo cambia su predicción. A pesar de que LIME ha mostrado su utilidad en ciertos casos, es imprescindible examinar más a fondo su utilidad real para incrementar la confianza de los usuarios en sistemas de visión artificial que se basan en DCNN.

Con el objetivo de contribuir en este campo de estudio, en este proyecto de investigación se realiza un análisis detallado del comportamiento del algoritmo LIME en una tarea simple de clasificación basado en imágenes naturales. Además, se desarrolló una interfaz web que, dada una imagen, muestre el resultado del clasificador y de LIME, lo cual permite a los usuarios analizar diferentes tipos de imágenes y observar la congruencia de la explicación de LIME con respecto a la manera en que los humanos explicaríamos dicha clasificación.

Índice

Índice de figuras

Lista de tablas

1 Capítulo 1: Introducción

La visión artificial es una de las primeras y mayores historias de éxito del aprendizaje profundo, el cual es uno de los paradigmas de aprendizaje más populares actualmente de la inteligencia artificial (IA). El modelo de aprendizaje profundo llamado redes neuronales convolucionales (DCNN por sus siglas en inglés) ha mostrado en los últimos años resultados notablemente buenos en los problemas de clasificación de imágenes y visión artificial. Sin embargo, el claro entendimiento de estos modelos y otros por parte de sus usuarios y diseñadores es una de las principales barreras a las que se enfrenta la IA en la actualidad para su implementación práctica. Un método para obtener cierta explicabilidad de una DCNN es LIME (por sus siglas del inglés: *Local Interpretable Model-agnostic Explanations* o Explicaciones Locales Independientes del Modelo) [1], el cual perturba la entrada a un sistema de clasificación a fin de observar como cambia su predicción. Aunque LIME ha mostrado ser útil para algunos casos, es necesario conocer más sobre su verdadera utilidad para aumentar la confianza de los usuarios de sistemas de visión artificial basados en DCNN.

A fin de aportar en esta área del conocimiento tan utilizada, en este proyecto de investigación se realiza un análisis detallado del comportamiento del algoritmo de explicabilidad LIME en un ejercicio simple pero claro de clasificación basado en imágenes naturales. Además, se construyó una interfaz web de presentación de resultados del clasificador y de los resultados de LIME que permite a otros usuarios hacer este análisis con diferentes tipos de imágenes.

1.1 Descripción del problema

El presente proyecto trata sobre el uso de las redes neuronales conocidas como profundas, para la creación de aplicaciones de visión computacional. A la fecha, es

inminente la necesidad de transparentar las decisiones que toman estos modelos, de forma que quienes los utilizan puedan decidir si deben confiar o no en su recomendación. Sin embargo, la transparencia no es una característica propia de las redes neuronales artificiales (RNA), por lo que es necesario construir otros algoritmos que permitan obtener algún tipo de explicación. En esta tesis se hace un análisis de la utilidad de uno de estos algoritmos, desde el punto de vista de un usuario común.

1.2 Justificación

El presente trabajo está dirigido a conocer a fondo el comportamiento del algoritmo LIME al usarse para explicar el resultado de una clasificación, permitiendo que los usuarios decidan hasta que punto se deben utilizar las predicciones del modelo de clasificación. La razón para realizar este proyecto es debido a que en la actualidad es inminente la necesidad de transparentar las decisiones que toman estos modelos de clasificación, de forma que quienes los utilizan puedan decidir si deben confiar o no en su recomendación.

1.3 Hipótesis

La explicación otorgada por el algoritmo de explicabilidad LIME, cuando se aplica en imágenes naturales de animales, tanto en escenarios complejos como simples y utilizando el clasificador basado en redes neuronales profundas, no es suficientemente parecida a la explicación comúnmente esperada por las personas.

1.4 Objetivo general

Analizar el comportamiento del algoritmo de explicabilidad de IA LIME en la clasificación de imágenes naturales de animales, al utilizarse en redes neuronales con-

volucionales profundas.

1.5 Objetivos particulares

1. Diseñar dos bases de datos de imágenes naturales de animales que permitan análisis simples de explicabilidad basada en LIME.

2. Programar un clasificador de imágenes naturales de animales usando transferencia de aprendizaje con la red DCNN conocida como VGG16 [2].

3. Evaluar cuantitativamente el comportamiento de explicabilidad de LIME al aplicarse en diferentes casos.

4. Construir una interfaz web de presentación de resultados del clasificador y de la explicabilidad de LIME sobre un servidor local.

1.6 Descripción de actividades

Para conseguir los objetivos de este proyecto se siguieron los siguientes pasos:

1. Revisión de teoría básica relacionada a la programación de la red VGG16, el algoritmo explicable LIME y varias técnicas de visualización de resultados.

2. Construcción de dos bases de datos de imágenes para realizar los experimentos.

3. Implementación en el ambiente de programación *Google colaboratory* de programas de clasificación, usando transferencia de aprendizaje sobre la red neuronal VGG16.

4. Análisis cuantitativo de las imágenes clasificadas por el clasificador.

5. Implementación en el ambiente de programación *Google colaboratory* de programas para generar la salida de explicación de LIME.

6. Aplicación de encuestas a usuarios comunes acerca de los rasgos físicos de los animales seleccionados que permiten distinguirlos entre sí.

7. Aplicación de encuestas a usuarios comunes para determinar su opinión sobre los resultados de la explicabilidad proporcionada por LIME.

8. Implementación de una interfaz web que dada una imagen permita visualizar el resultado del clasificador y de la interpretación producida por LIME.

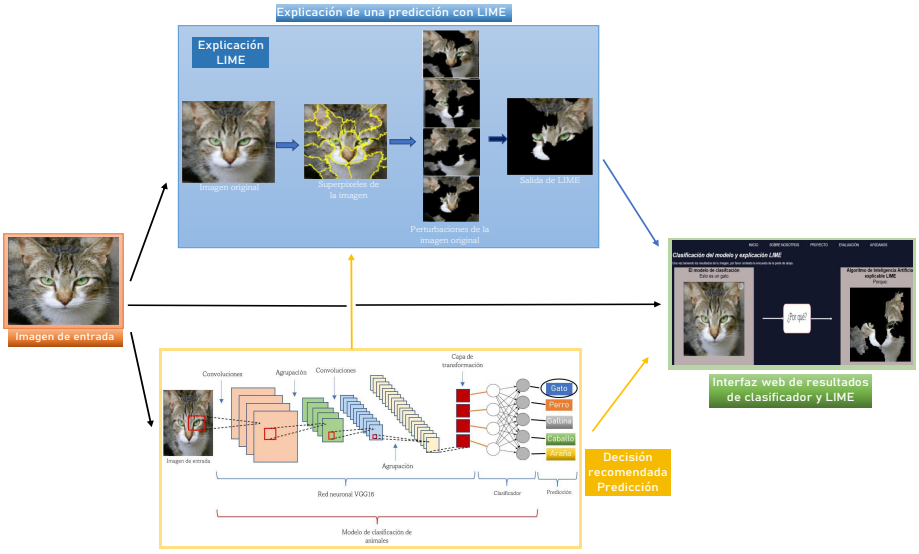
9. Recopilación descriptiva de las encuestas evaluadas y análisis de suficiencia de tamaño de muestras, distribución de frecuencias de respuestas y porcentajes de la congruencia de LIME con respecto a lo esperado por las personas.

10. Análisis de la experiencia del usuario utilizando la interfaz mediante la aplicación del cuestionario SUS (*System Usability Scale*) [3].

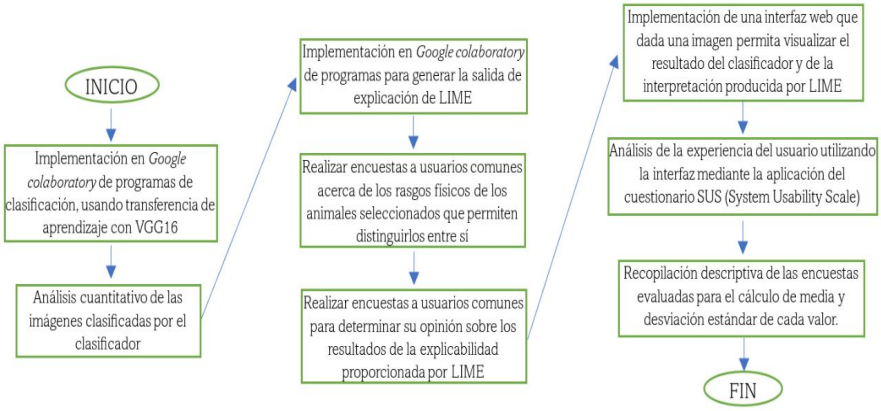
11. Comparar la usabilidad de la interfaz en el subconjunto de muestras representando la comunidad que pertenece a la Facultad de Ciencias de la Electrónica (FCE) y la comunidad que no pertenece a la FCE mediante la prueba estadística t.

1.7 Diagrama de bloques

En la Fig. 1(a) se muestra, de forma general, un diagrama de las entradas y salidas del sistema base para la realización de esta investigación. Por otra parte, la Fig. 1(b) detalla las actividades que se desarrollaron durante esta investigación.



(a) Diagrama a bloques



(b) Flujo de actividades

Figura 1: Diagrama del proyecto

1.8 Contribuciones

Este trabajo de investigación tiene las siguientes contribuciones:

1. Un análisis cualitativo de la explicabilidad percibida por usuarios del algoritmo de Inteligencia Artificial Explicable (XAI por sus siglas del inglés) LIME, con respecto a su propia intuición en la clasificación de imágenes naturales de animales.

2. Un sistema de clasificación, ejecutable a través de una interfaz web, para clasificación y explicación automática de imágenes naturales de animales, usando el *framework* VGG16 como clasificador y el algoritmo LIME como explicación.

3. Dos bases de datos, depuradas para el estudio de modelos explicables en imágenes naturales de 5 clases de animales.Una base de datos con imágenes en escenarios simple y una base de datos con imágenes en escenarios complejos.

1.9 Organización del documento

El presente trabajo se estructura de la siguiente manera. En primer lugar, se presenta la introducción, que incluye la descripción del problema, los objetivos, la justificación, descripción de actividades e hipótesis de la investigación. Además, se proporciona una descripción detallada de la estructura del documento.
El segundo capítulo está dedicado al marco teórico. En esta sección, se realiza una revisión exhaustiva de los conceptos teóricos y fundamentos clave para la investigación. Asimismo, se brinda una explicación detallada de aquellos aspectos que son relevantes y necesarios para comprender el contexto en el que se desarrolla el trabajo.
En el tercer capítulo, se lleva a cabo un análisis de los estudios e investigaciones previas que se relacionan directamente con el tema de investigación, y se destacan

los avances y contribuciones en el campo de estudio de esta investigación.

En el cuarto capítulo, se describe en detalle el sistema utilizado para este trabajo. Se presentará la metodología propuesta, así como los componentes, técnicas y herramientas utilizadas para el desarrollo del sistema.

El quinto capítulo presenta los resultados obtenidos a partir del análisis de los datos recopilados, los cuales se presenta utilizando gráficos y tablas para facilitar su comprensión.

Finalmente, en el capítulo de conclusiones y trabajo futuro, se lleva a cabo una discusión detallada de los resultados obtenidos. Por otra parte, se exploran las limitaciones del estudio y las posibles investigaciones futuras que puedan abordar las áreas pendientes de esta investigación.

Además de los capítulos mencionados, se incluyen algunos apéndices que contienen información adicional relevante, como cuestionarios y códigos de programación utilizados.

2 Capítulo 2: Marco Teórico

2.1 Inteligencia Artificial, Inteligencia Computacional y Aprendizaje de Máquina

Según la Organización para la Cooperación y el desarrollo Económicos (OECD, por sus siglas del francés), un sistema de Inteligencia Artificial (IA) es aquel basado en una máquina capaz de influenciar el medio ambiente, produciendo una salida (recomendaciones, predicciones o decisiones) para un conjunto de objetivos dados [4]. Éste se caracteriza porque:

1. Percibe el medio ambiente a través de datos o entradas.
2. Abstrae estas percepciones hacia modelos.
3. Usa los modelos para formular opciones de salidas.

La Inteligencia Computacional (IC) es el área de la IA que incluye conceptos, algoritmos y marcos de trabajo, los cuales permiten desarrollar aplicaciones que exhiben un comportamiento inteligente y son útiles en sistemas que resultan complejos para otro tipo de aplicaciones. Las soluciones basadas en IC se caracterizan porque sus algoritmos son de tipo sub-simbólico y muchas veces están inspirados en la naturaleza [5].
Aprendizaje de Máquina (AM) es la capacidad de un dispositivo de adaptarse a nuevas circunstancias y extrapolar patrones encontrados [6].

2.2 Neurona biológica

El procesamiento de información realizado por el cerebro humano se lleva a cabo mediante componentes de procesamiento biológico, que operan en paralelo, para

producir funciones como pensar y aprender. La célula fundamental del sistema nervioso central es la neurona (Fig. 2), y su función se reduce a conducir impulsos (estímulos eléctricos originados por reacciones físico-químicas) en determinadas condiciones de funcionamiento. Este componente biológico se puede dividir en tres partes principales: dendritas, cuerpo celular (también conocido como "soma") y axón.

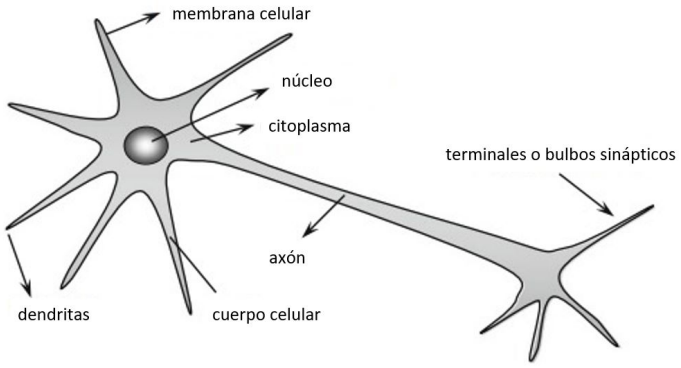


Figura 2: Neurona biológica simple [7]

Las dendritas se componen de varias extensiones delgadas que forman el árbol dendrítico. El propósito fundamental de las dendritas es adquirir, continuamente, estímulos de varias otras neuronas (conectores) o del ambiente externo. El cuerpo celular es el encargado de procesar toda la información que proviene de las dendritas, para producir un potencial de activación que indica si la neurona puede desencadenar un impulso eléctrico a lo largo de su axón. Es también en el cuerpo celular donde se encuentran los principales orgánulos citoplasmáticos (núcleo, mitocondrias, centriolo, lisosoma, etc.) de la neurona. El axón está compuesto por una sola extensión cuya misión es guiar los impulsos eléctricos a otras neuronas de conexión, o a las neuronas directamente conectadas al tejido muscular (neuronas eferentes). La terminación del axón también se compone de ramas llamadas terminales sinápticas. Las sinapsis son las conexiones que permiten la transferencia de impulsos de axones eléctricos de una neurona particular a las dendritas de otras neuronas, cómo se ilus-

tra en la Fig. 3. Es importante tener en cuenta que no hay contacto físico entre las neuronas que forman la unión sináptica, por lo que los elementos neurotransmisores liberados en la unión son los encargados de ponderar la transmisión de una neurona a otra [8].

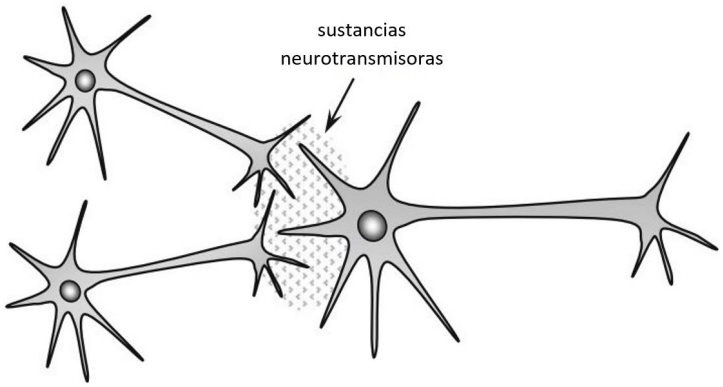


Figura 3: Ilustración de la conexión sináptica entre neuronas [8]

2.3 Neurona artificial

Los componentes computacionales o unidades de procesamiento, llamadas neuronas artificiales, son modelos simplificados de las neuronas biológicas. Estos modelos fueron inspirados por el análisis del proceso que se lleva a cabo en una membrana celular a fin de que una neurona genere y propague impulsos eléctricos.

Las neuronas artificiales usadas en RNA son no lineales, generalmente proporcionan salidas continuas y realizan funciones simples, como recopilar señales disponibles en sus entradas, ensamblarlas de acuerdo con sus funciones operativas y producir una respuesta considerando sus funciones de activación innatas.

El modelo más simple que incluye las principales características de una red neuronal biológica (paralelismo y alta conectividad), fue propuesta por McCulloch y Pitts en 1943 [9], y sigue siendo el modelo más usado en distintas arquitecturas de RNA [7]. La Fig. 4 muestra como puede ser representada una neurona artificial. Las múltiples señales de entrada provenientes de un entrono externo (aplicación) son representadas

por el conjunto $\{x_1, x_2, x_3, \ldots, x_n\}$ análogo a los impulsos eléctricos externos recogidos por las dendritas en la neurona biológica. El pesaje realizado por las uniones sinápticas de la red se implementan sobre la neurona artificial como un conjunto de pesos sinápticos $\{w_1, w_2, w_3, \ldots, w_n\}$. Análogamente, la relevancia de cada una de las entradas neuronales $\{x_i\}$ se calcula multiplicándolas por su correspondiente peso sináptico $\{w_i\}$, ponderando así toda la información externa que llega a la neurona. Por lo tanto, la salida del cuerpo celular artificial, denotada por u es la suma ponderada de sus entradas.

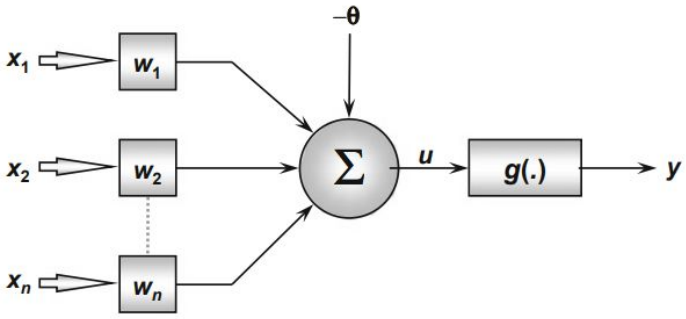


Figura 4: Neurona artificial simple [8]

La neurona artificial se compone de 7 elementos básicos:

1. Señales de entrada $(x_1, x_2, \ldots, x_n)$ son las señales o muestras procedentes del entorno externo y representan los valores asumidos por las variables de una aplicación determinada. Las señales de entrada generalmente se normalizan para mejorar la eficiencia computacional de los algoritmos de aprendizaje.

2. Pesos sinápticos $(w_1, w_2, \ldots, w_n)$ son los valores utilizados para ponderar cada una de las variables de entrada, lo que permite cuantificar su relevancia con respecto a la funcionalidad de la neurona.

3. El sumador lineal $(\sum)$ reúne todas las señales de entrada ponderadas por los pesos sinápticos para producir un voltaje de activación.

4. El umbral de activación (θ) es una variable utilizada para especificar el umbral adecuado que debe tener el resultado producido por el sumador lineal para generar un valor de activación hacia la salida de la neurona.

5. El potencial de activación (u) es el resultado producido por la diferencia entre el sumador lineal y el umbral de activación.

6. La función de activación (g) cuyo objetivo es limitar la salida de la neurona dentro de un rango razonable de valores, asumido por su propia funcionalidad.

7. La señal de salida (y), que consiste en el valor final producido por la neurona dado un conjunto particular de señales de entrada, y también se puede utilizar como entrada para otras neuronas interconectadas secuencialmente

Las siguientes expresiones sintetizan el resultado producido por la neurona artificial propuesta por McCulloch y Pitts:

$$u = \sum_{i=1}^{n} w_i x_i - \theta \tag{1}$$

$$y = g(u) \tag{2}$$

2.4 Red neuronal artificial

Las RNA son modelos matemáticos inspirados en sistemas biológicos, adaptados y simulados en computadoras convencionales. Los componentes computacionales de las RNA , llamadas neuronas artificiales, son modelos simplificados de la neurona biológica. Una RNA consiste en capas de neuronas artificiales conectadas unas a otras. Existen un gran número de formas para conectarlas, lo que origina diferentes modelos de RNA con diferentes capacidades.
Generalmente, una red neuronal artificial está formada por capas, las cuales pueden dividirse en 3 partes:

- Capa de entrada, la cual es la responsable de recibir algún tipo de información, por ejemplo datos o señales crudas, características o medidas de un entorno externo.

- Capas ocultas o capas invisibles, compuestas por neuronas que se encargan de extraer patrones asociados al proceso o sistema que se está analizando. Estas capas realizan la mayor parte del procesamiento interno desde una red.

- Capa de salida, compuesta por neuronas, responsables de producir y presentar las salidas finales de la red, que resultan del procesamiento realizado por las neuronas en las capas anteriores.

Las características principales de las RNA son:

- Aprendizaje: Una red neuronal puede modificar su comportamiento en respuesta al medio ambiente.

- Generalización: Una vez entrenada, la red neuronal puede ser insensible a cambios en sus entradas.

- Abstracción: Una red neuronal puede determinar la esencia o características principales de un conjunto de datos.

2.5 Proceso de entrenamiento y propiedades del aprendizaje

Una de las características más relevantes de las RNA es su capacidad de aprendizaje a partir de la presentación de muestras (patrones), que expresa el comportamiento del sistema. Por lo tanto, después de que la red ha aprendido la relación entre entradas y salidas, puede generalizar soluciones, lo que significa que la red puede producir una salida que está cerca de la salida esperada (o deseada) de cualquier valor de entrada dado.

El proceso de entrenamiento de una red neuronal consiste en aplicar los pasos ordenados necesarios para ajustar los pesos sinápticos y los umbrales de sus neuronas, con el fin de generalizar las soluciones producidas por sus salidas. El conjunto de pasos ordenados utilizados para entrenar la red se llama algoritmo de aprendizaje. Durante su ejecución, la red podrá extraer características discriminantes sobre el sistema que se está mapeando a partir de muestras adquiridas del sistema.

Por lo general, el conjunto completo que contiene todas las muestras disponibles del comportamiento del sistema se divide en dos o tres subconjuntos, que se denominan subconjunto de entrenamiento, subconjunto de validación y subconjunto de prueba. El subconjunto de entrenamiento, comúnmente compuesto por un $60-90\%$ de muestras tomadas de forma aleatoria del conjunto completo, se utiliza en el proceso de aprendizaje. Por otro lado, el subconjunto de prueba, que se compone del $10-40\%$ del conjunto de muestra completo, se utilizará para evaluar si las capacidades de la red de generalizar soluciones están dentro de niveles aceptables. El subconjunto de validación, es utilizado cuando se dispone de datos suficientes, a fin de ajustar los hiper-parámetros del modelo sin generar sesgos producidos al usar varias veces el conjunto de pruebas. Es importante aclarar que al dimensionar estos subconjuntos se deben considerar las características estadísticas de los datos.

Durante el proceso de entrenamiento de RNA, cada presentación completa de todas las muestras pertenecientes al conjunto de entrenamiento, con el fin de ajustar los pesos y umbrales sinápticos, se denominará época de entrenamiento [8].

2.5.1 Aprendizaje supervisado

La estrategia de aprendizaje supervisado consiste en tener disponibles las salidas deseadas para un conjunto dado de señales de entrada, que representan el comportamiento del modelo; en otras palabras, cada muestra del conjunto de entrenamiento está compuesta por las señales de entrada y sus correspondientes salidas.

Es a partir de esta información que las estructuras neuronales formularán "hipótesis" sobre el sistema que se está aprendiendo. La aplicación del aprendizaje supervisado solo depende de la disponibilidad de la tabla de atributos/valores, y se comporta como si un "entrenador" estuviera enseñando a la RNA cuál es la respuesta correcta para cada muestra presentada para su entrada.
Los pesos y umbrales sinápticos de la red se ajustan continuamente mediante la aplicación de acciones comparativas, ejecutadas por el propio algoritmo de aprendizaje, que supervisan la discrepancia entre las salidas producidas con respecto a las salidas deseadas, utilizando esta diferencia en el procedimiento de ajuste. La red se considera "entrenada" cuando esta discrepancia se encuentra dentro de un rango de valores aceptable, teniendo en cuenta los propósitos de generalización de soluciones [8].

2.6 Alcances y limitaciones de las RNA

- Las RNA no son la solución de todos los problemas, sino solo de aquellos en los que "las reglas de solución" no son conocidas, y existen suficientes datos ejemplos que permitan a la red aprender.
- Las RNA son hasta cierto punto impredecibles.
- Las RNA no pueden explicar como resuelven un problema. La representación interna generada puede ser demasiado compleja para ser analizada, aún y en los casos más sencillos. [10].

2.7 Perceptrón multicapa

El perceptrón multicapa (MLP por sus siglas en inglés) o también conocido como red neuronal densa, presenta al menos una capa neuronal intermedia (oculta), que se colocan entre la capa de entrada y la capa de salida respectiva. En consecuencia, las redes MLP tienen al menos dos capas neuronales, y sus neuronas se distribuyen

entre las capas intermedias y de salida.

El MLP también es conocido por su amplia gama de aplicación en varios problemas de diferentes áreas de conocimiento y también se considera una de las arquitecturas más versátiles en cuanto a aplicabilidad. Normalmente el entrenamiento de las MLP se realiza con un proceso supervisado. En la Fig. 5 se observa que las entradas de la red, que representa las señales de una aplicación determinada, se propagarán capa por capa hacia la capa de salida. En este caso, las salidas de las neuronas de la primera capa neuronal serán las entradas de las neuronas de la segunda capa neuronal oculta. La propagación de las señales de entrada de una red neuronal densa, independientemente del número de capas intermedias, siempre fluye en una dirección, es decir, desde la capa de entrada a la capa neuronal de salida [7].

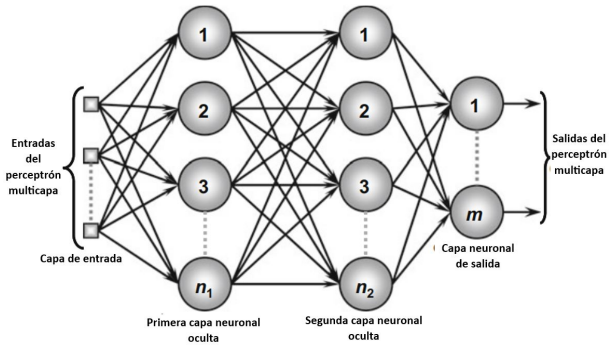


Figura 5: Ilustración del perceptrón multicapa [7]

2.7.1 Proceso de entrenamiento del perceptrón multicapa

El proceso de entrenamiento de las MLP puede realizarse de muchas maneras, pero en general se utiliza el algoritmo de retropropagación, el cual es realizado mediante la aplicación sucesiva de dos etapas específicas. Estás etapas se ilustran en la Fig. 6, que muestra una configuración compuesta por dos capas ocultas, n señales en la capa de entrada, n_1 neuronas en la primer capa oculta, n_2 neuronas en la segunda capa oculta y n_3 señales asociadas con la capa de salida(tercera capa neuronal).

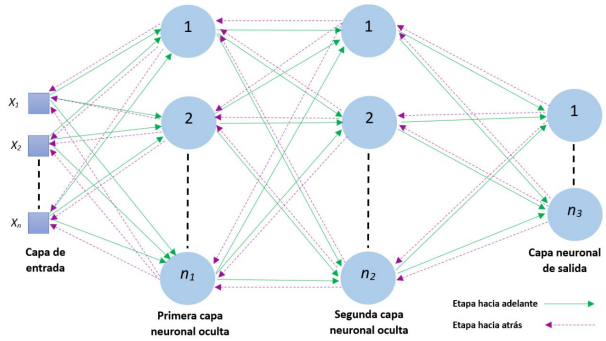


Figura 6: Ilustración de ambas etapas de formación del perceptrón multicapa [7]

La primera etapa se llama propagación hacia adelante, donde las señales $\{x_1, x_2, ..., x_n\}$ de una muestra dada del conjunto de entrenamiento se insertan en las entradas de la red y se propagan capa por capa hasta la producción de las salidas correspondientes. De este modo, esta etapa pretende únicamente obtener las respuestas de la red, teniendo en cuenta solo los valores actuales de los pesos sinápticos y los umbrales de sus neuronas, que permanecerán sin modificaciones durante la ejecución de esta etapa.

A continuación, las respuestas producidas por los resultados de la red se comparan con las respectivas respuestas deseadas, ya que es un proceso de aprendizaje supervisado. Considerando una red densa con n_3 neuronas en su capa de salida, se calculan las respectivas n_3 desviaciones (errores) entre las respuestas deseadas y las producidas por las neuronas de salida y se utilizarán después para ajustar los pesos y umbrales de todas las neuronas.

Por lo tanto, debido a estos errores, se aplica la segunda etapa del algoritmo de retropropagación, conocida como propagación hacia atrás. A diferencia de la primera etapa, las modificaciones (ajustes) de los pesos sinápticos y umbrales de todas las neuronas de la red se ejecutan durante esta etapa. Para una descripción detalla del algoritmo de retropropagación, ver [11].

2.8 Red Neuronal Convolucional

Este tipo de red está inspirada en la estructura de la retina humana, en la que las neuronas sensoriales tienen un llamado campo receptivo, es decir, una región limitada en la que responden a un estímulo (visual) [5]. La figura 7 muestra un ejemplo de este tipo de redes.

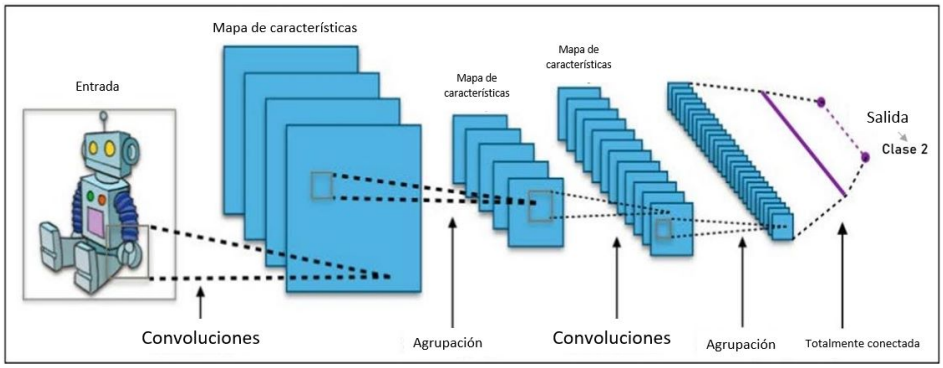


Figura 7: Ejemplo de una Red Neuronal Convolucional [12]

Las propiedades que diferencian una red neuronal convolucional (RNC) de otras RNA son:

1. Los patrones que aprenden son invariantes a traslaciones amplias: Una vez que aprende un patrón lo puede reconocer en cualquier lugar, mientras que una MLP tiene que conocer un nuevo patrón si un pequeño patrón cambia de lugar.

2. Aprenden por jerarquías espaciales de patrones: La primer capa aprende pequeños patrones locales, la segunda aprende patrones de las características de la primer capa (Fig. 8). Esto permite que las RNC aprendan de manera eficiente conceptos visuales cada vez más complejos y abstractos, porque el mundo visual es fundamentalmente espacialmente jerárquico.

3. En cualquier capa, las neuronas tienen los mismos parámetros de peso en toda la capa.

4. Las RNC abandonan la antigua función de salida sigmoidal y, en su lugar, utilizan la unidad lineal rectificada (ReLU), que también es una función no lineal.

5. Algunos modelos intercalan capas de convolución con capas de submuestreo o "agrupación".

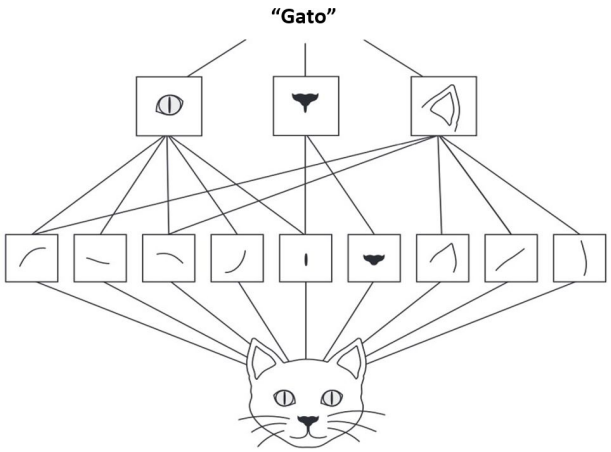


Figura 8: El mundo visual forma una jerarquía espacial de módulos visuales: Líneas elementales o texturas se combinan en objetos simples como ojos u orejas, que se combinan en conceptos de alto nivel como "gato". [13]

Las RNC usan el aprendizaje supervisado y entrenan mediante retropropagación.
Una red del tipo general RNA multiplica cada entrada por un peso ya determinado y las suma. ReLU significa Unidad lineal rectificada. Esta es una función no lineal definida por $max\ (0, x)$ (Fig. 9), dónde x es el valor de salida de la capa de convolución inmediatamente anterior. La Fig. 9 muestra la gráfica de esta función [14].

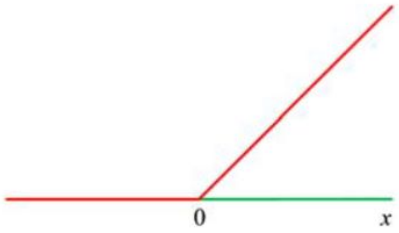


Figura 9: La función no lineal ReLU [14]

2.8.1 Funcionamiento de las convoluciones

Las convoluciones operan sobre tensores 3D llamados mapas de características, con dos ejes espaciales (altura y anchura), así como un eje de profundidad (también llamado eje de canales). Para una imagen RGB, la dimensión del eje de profundidad es 3, porque la imagen tiene tres canales de color: rojo, verde y azul. Para una imagen en blanco y negro, la profundidad es 1 (niveles de gris). La operación de convolución extrae parches de su mapa de características de entrada y aplica la misma transformación a todos estos parches, produciendo un mapa de características de salida. Este mapa de características de salida sigue siendo un tensor 3D, tiene altura y anchura. La profundidad es arbitraria ya que es un parámetro de la capa, y los diferentes canales en ese eje de profundidad ya no representan colores específicos como en la entrada RGB; Más bien, representan filtros. Los filtros codifican aspectos específicos de los datos de entrada.
Las convoluciones son definidas por dos parámetros:

- Tamaño de los parches extraídos de las entradas: normalmente son 3 × 3 o 5 × 5.
- Profundidad del mapa de características de salida: este es el número de filtros calculados por la convolución.

Una convolución funciona deslizando estas ventanas de tamaño 3 × 3 o 5 × 5 so-

bre el mapa de características de entrada 3D, deteniéndose en cada ubicación posible y extrayendo el parche 3D de las características circundantes (forma (alto_ventana, ancho_ventana, profundidad_entrada)). Cada parche 3D de este tipo se transforma luego en un vector de forma 1D (profundidad de salida), que se realiza a través de un producto tensorial con una matriz de peso aprendida, llamada kernel de convolución: el mismo kernel se reutiliza en cada parche. Todos estos vectores (uno por parche) se vuelven a ensamblar espacialmente en un mapa de salida 3D de la forma (altura, anchura, profundidad de salida). Cada ubicación espacial en el mapa de características de salida corresponde a la misma ubicación en el mapa de características de entrada (por ejemplo, la esquina inferior derecha de la salida contiene información sobre la esquina inferior derecha de la entrada). El proceso completo se muestra en la Fig. 10.

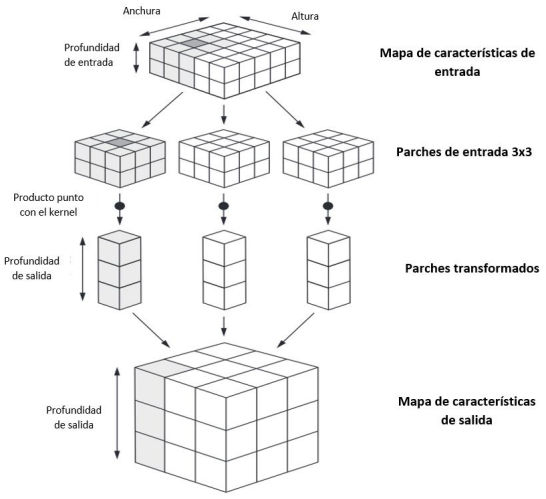


Figura 10: Funcionamiento de una convolución [13]

Un factor que puede influir en el tamaño de salida es el parámetro paso; la distancia entre dos ventanas sucesivas es un paso, por defecto es 1.

2.8.2 Operación agrupación máxima

La agrupación máxima consiste en extraer ventanas de los mapas de características de entrada y generar el valor máximo de cada canal. Es conceptualmente similar a la convolución, excepto que en lugar de transformar parches locales a través de una transformación lineal aprendida (el kernel de convolución), se transforman a través de una operación codificada de tensor máximo. Una gran diferencia con la convolución es que la agrupación máxima generalmente se realiza con ventanas de 2 $\times$ 2 y paso 2, para reducir la muestra de los mapas de características en un factor de 2. Por otro lado, la convolución generalmente se realiza con ventanas de 3 $\times$ 3 Y sin paso (paso 1) [13].

La capas de agrupación tienen una ventaja sobre las capas convolucionales, puesto que son más fáciles y rápidas de calcular, ya que reduce los cálculos y sólo se trata de buscar los máximos y extrae la característica más nítida de la imagen [15].

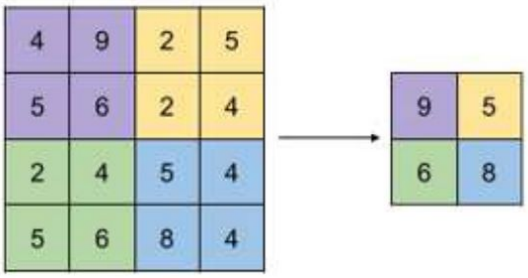


Figura 11: Ejemplo de la agrupación máxima [16]

2.9 Técnicas usadas para reducir el sobreentrenamiento(*overfiting*)

2.9.1 ¿Qué es el sobreentrenamiento?

En el aprendizaje automático supervisado, hay un problema fundamental. Cuando el modelo no generaliza bien de los datos del entrenamiento a los datos desconocidos, se denomina sobreentrenamiento o sobreajuste. Debido al sobreentrenamiento,

el modelo funciona perfectamente en el conjunto de entrenamiento, mientras que en el conjunto de prueba tiene un mal desempeño.
Una de las causas principales de este fenómeno es el ruido en el conjunto de entrenamiento; también se presenta cuando el conjunto de entrenamiento es demasiado pequeño en tamaño, o tiene menos datos representativos [17]. En la práctica, los datos siempre se ven afectados por el ruido, es decir, contienen errores aleatorios y desviaciones de los valores verdaderos, que pueden pertenecer tanto a las entradas como a las salidas [5].

2.9.2 Técnica de *dropout*

El *dropout* es una de las técnicas más eficaces y más utilizados para las RNA, que se puede aplicar a la salida de algunas de las capas de la red. Aplicada a una capa, ésta técnica consiste en ajustar a cero aleatoriamente una serie de características de salida de la capa durante el entrenamiento. Por lo tanto, si una capa dada normalmente devolvería un vector [0.2, 0.5, 1.3, 0.8, 1.1] para una muestra de entrada durante el entrenamiento. Después de aplicar *dropout*, este vector tendrá unas pocas entradas ajustadas a cero distribuidas al azar: por ejemplo, [0, 0.5, 1.3, 0, 1.1]. La tasa de *dropout* es la fracción de las características que se ajustan a cero; Por lo general, se establece entre 0.2 y 0.5. La idea central es que la introducción de ruido en los valores de salida de una capa puede romper los patrones de casualidad que no son significativos, que el modelo comenzará a memorizar si no hay ruido presente [13]. El *dropout* se puede aplicar después de capas convolucionales, agrupadas o totalmente conectadas [18]. En la Fig. 12, se puede observar un ejemplo para capas totalmente conectadas.

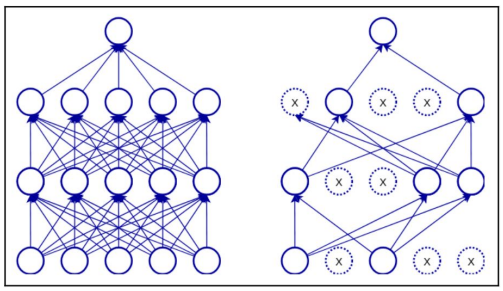


Figura 12: Un ejemplo de *dropout* en capas totalmente conectadas [18]

2.9.3 Técnica de aumento de datos

Dado que una de las causas del sobreentrenamiento es por tener muy pocas muestras de las que aprender, lo que le impide entrenar un modelo que pueda generalizarse nuevos datos. El aumento de datos adopta el enfoque de generar más datos de entrenamiento a partir de muestras de entrenamiento existentes al aumentar las muestras a través de una serie de transformaciones aleatorias que producen imágenes de aspecto creíble. El objetivo es que, en el momento del entrenamiento, el modelo nunca vea exactamente la misma imagen dos veces. Esto ayuda a exponer el modelo a más aspectos de los datos para que pueda generalizarse mejor.

Las tranformaciones utilizadas son:

- RandomFlip ("horizontal" o "vertical"). Aplica un volteo horizontal o vertical aleatoriamente a un 50% de las imágenes de entrenamiento.
- RandomRotation(x). Gira las imágenes de entrada en un valor aleatorio en el rango de $[-x\ \%,+x\ \%]$(éstas son fracciones de un círculo completo; en grados, el rango sería $[-x * 360$ grados, $+x * 360$ grados$]$)).
- RandomZoom(x). Acerca o aleja la imagen mediante un factor aleatorio en el rango $[-x\ \%,+x\ \%]$.

En la Fig. 13 se muestra un ejemplo del aumento de datos, en la cual en la parte (b) se muestran las transformaciones de la parte (a).

Figura 13: Generación de variaciones de una imagen a través del aumento de datos aleatorios [13]

Si entrenamos un nuevo modelo utilizando esta configuración de aumento de datos, el modelo nunca verá la misma entrada dos veces. Pero las entradas que ve todavía están fuertemente inter-correlacionadas porque provienen de un pequeño número de imágenes originales. Por lo tanto, esto no es suficiente para deshacerse por completo del sobreentrenamiento [13].

2.9.4 Transferencia de aprendizaje

La extracción de características consiste en utilizar las representaciones aprendidas por un modelo previamente entrenado para extraer características interesantes de nuevas muestras. Estas características se ejecutan a través de un nuevo clasificador, que se entrena desde cero.

Las RNC utilizadas para la clasificación de imágenes comprenden dos partes: comienzan con una serie de capas de agrupación y convolución, y terminan con un clasi-

ficador totalmente conectado. La primera parte se llama base convolucional del modelo. En el caso de las RNC, la extracción de características consiste en tomar la base convolucional de una red previamente entrenada, ejecutar los nuevos datos a través de ella y entrenar un nuevo clasificador sobre la salida. En general solo se rehúsa la base convolucional y no el clasificador, ya que las representaciones aprendidas por la base convolucional probablemente sean más genéricas y, por lo tanto, más reutilizables: los mapas de características de una RNC son mapas de presencia de conceptos genéricos sobre una imagen, que probablemente sean útiles independientemente del problema de visión por computadora en cuestión. Pero las representaciones aprendidas por el clasificador serán necesariamente específicas para el conjunto de clases en las que se entrenó el modelo [13].

La extracción de características con aumento de datos, es costosa computacionalmente, pero nos permite crear un modelo que encadene la base convolucional con un nuevo clasificador denso, y entrenarlo de extremo a extremo en las entradas [13]. Para hacer esto, primero se congela la base convolucional. Congelar una capa o conjunto de capas significa evitar que sus pesos se actualicen durante el entrenamiento. Si no hacemos esto, las representaciones que fueron aprendidas previamente por la base convolucional (del modelo preentrenado) se modificarán durante el entrenamiento. Debido a que las capas densas en la parte superior se inicializan aleatoriamente, las actualizaciones de peso muy grandes se propagarían a través de la red, destruyendo efectivamente las representaciones aprendidas previamente [13].

El siguiente paso es realizar un ajuste fino de un modelo preentrenado, el cual consiste en descongelar algunas de las capas de la base convolucional congelada del modelo preentrenado utilizado para la extracción de características y entrenar conjuntamente la parte recién agregada del modelo (en este caso, el clasificador totalmente conectado) y estas capas superiores. Esto se llama ajuste fino porque ajusta ligeramente las representaciones más abstractas del modelo que se está reutilizando para hacerlas más relevantes para el problema en cuestión. Es posible actualizar todas las neuronas de la red. Puesto que, las capas iniciales detectan características

generales, no relacionadas con una tarea específica, tiene sentido reutilizarlas. Por otro lado, las capas más profundas pueden detectar características específicas de la tarea y sería mejor actualizarlas [18].

2.10 Matriz de confusión y exactitud de un modelo entrenado

La matriz de confusión es una de las formas más populares de evaluar un modelo de clasificación. Aunque la matriz por sí misma no es una métrica, la representación de la matriz se puede utilizar para definir una variedad de métricas, las cuales se vuelven importantes en algún caso o escenario específico. Se puede crear una matriz de confusión para una clasificación binaria, así como para un modelo de clasificación multiclase [19].

Una matriz de confusión se crea comparando la predicción de salida del modelo con la clase real del objeto de prueba. Esta comparación se repite para todo el conjunto de prueba y los resultados de esta comparación se compilan en formato de matriz o tabla. Esta matriz resultante es la matriz de confusión. Por lo tanto, la matriz de confusión es una estructura tabular para realizar un seguimiento de las clasificaciones correctas, así como las clasificaciones erróneas. La Fig. 14 muestra un ejemplo de una matriz de confusión, donde puede apreciarse el ordenamiento de renglones y columnas que permite calcular varias métricas del desempeño y calidad de un clasificador, entre ellas la exactitud.

La exactitud es una de las medidas más populares del rendimiento del clasificador. Se define como la precisión general o proporción de predicciones correctas del modelo [19]. Mediante la ecuación 3. Utilizando los valores de la matriz de confusión se puede calcular la exactitud del modelo (Ecuación 3)

$$Exactitud = \frac{VN + VP}{VN + VP + FP + FN} \tag{3}$$

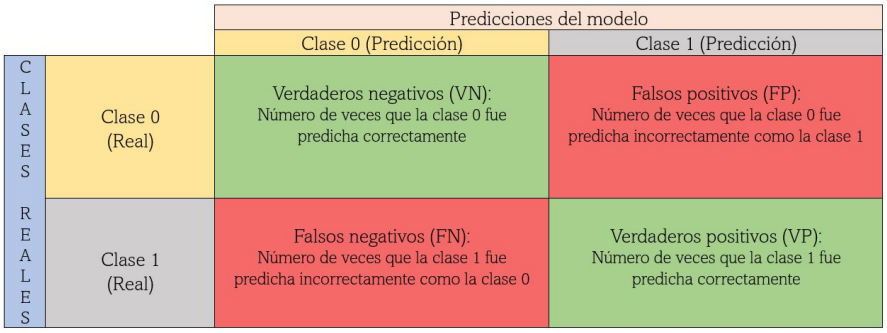


Figura 14: Estructura típica de una matriz de confusión [19]

2.11 Modelo preentrenado VGG16

En el contexto de redes neuronales convolucionales, un modelo pre-entrenado es un modelo que se entrenó previamente en un gran conjunto de datos, generalmente en una tarea de clasificación de imágenes a gran escala. Si este conjunto de datos original es lo suficientemente grande y general, la jerarquía espacial de características aprendidas por el modelo preentrenado puede actuar efectivamente como un modelo genérico del mundo visual y, por lo tanto, sus características pueden resultar útiles para muchos problemas diferentes de visión por computadora, aunque estos nuevos problemas pueden involucrar clases completamente diferentes a las de la tarea original. Tal portabilidad de las características aprendidas a través de diferentes problemas es una ventaja clave del aprendizaje profundo en comparación con muchos enfoques de aprendizaje superficiales más antiguos, y hace que el aprendizaje profundo sea muy efectivo para problemas de datos pequeños [13].

La arquitectura VGG16 es una de estas arquitecturas pre-entrenadas, y fue publicada por Karen Simonyan y Andrew Zisserman en 2014 [2]. Las siglas VGG significan *Visual Geometry Group* y presenta una arquitectura de 16 capas, que incluyen capas convolucionales, capas *max-pooling* y capas de activación; las capas están conectadas totalmente. VGG16 es uno de los modelos más investigados actualmente.

2.11.1 Arquitectura de la red VGG16

El modelo VGG16 se compone de bloques de capas convolucionales, que tienen 2 o 3 convoluciones. Utiliza 13 capas convolucionaes. Las capas convolucionales de la red VGG16 son en su mayor parte filtros de tamaño 3×3 con un tamaño de paso de 1, y las capas de agrupación son de 2 × 2 con un tamaño de paso de 2 (Fig. 15). Una pila de capas convolucionales es seguida por tres capas totalmente conectadas: las dos primeras tienen 4096 neuronas cada una, la tercera realiza una clasificación, la cual consta de 1000 neuronas y, por lo tanto, contiene 1000 canales (uno para cada clase). La capa final es la capa *softmax*. La configuración de las capas totalmente conectadas es la misma en todas las redes [2]- [15]. La Fig. 15 muestra la arquitectura de VGG16.

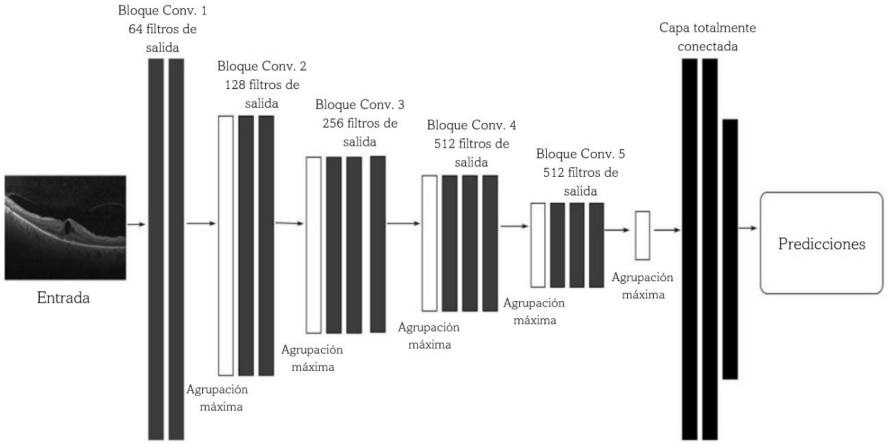


Figura 15: Arquitectura de la red VGG16 [15]

2.12 Inteligencia Artificial explicable (XAI)

La inteligencia artificial explicable (XAI, por sus siglas en inglés) se puede definir de la siguiente forma: *Dada una audiencia, una inteligencia artificial explicable es aquella que produce detalles o razones que permiten que su funcionamiento* sea claro o fácil de entender [20]. Esta definición ubica a la audiencia (usuarios) como el

aspecto clave a considerar al explicar el modelo. La Fig. 16, publicada por [20], muestra los diferentes tipos de usuarios de un sistema basado en IA y la manera en que pueden beneficiarse de un sistema XIA.

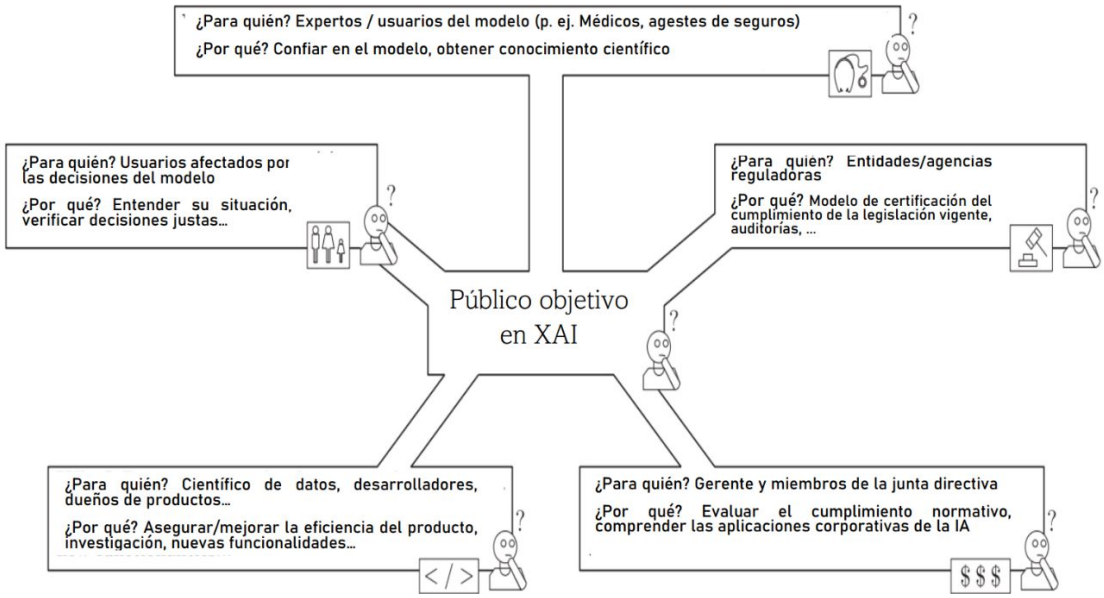


Figura 16: Diagrama que muestra los diferentes propósitos de la explicabilidad en los modelos ML buscados por diferentes perfiles de audiencia. Sin embargo, cualquier medio útil para reducir la complejidad de un modelo o para simplificar sus resultados puede considerarse como un enfoque en XAI [20]

En los últimos años, XAI ha tenido una gran atención debido al crecimiento exponencial en el uso de técnicas de aprendizaje automático para la toma de decisiones en la práctica diaria, lo que ha provocado la necesidad de que los usuarios se aseguren de que las recomendaciones que les está dando un sistema automático, sean realmente confiables.

Un problema importante en XAI es la confusión que existen en cuanto a varios términos comúnmente utilizados y a veces con interpretaciones contradictorias, por lo que es fundamental definir la terminología que se usará al respecto en este proyecto, la cual fue tomada de [20] y se describe a continuación.

- Algoritmos de caja negra: Aquellos cuyo funcionamiento no es directamente claro para sus usuarios. Cuando se usan algoritmos de caja negra los usuarios deben buscar formas para descubrir su lógica interna [21].

- Interpretabilidad: Característica pasiva de un modelo que se refiere al nivel en el que este modelo, por sí mismo, hace sentido a un observador humano. Muchas veces se utiliza esta palabra como sinónimo de "transparencia".

- Explicabilidad: Característica activa de un modelo que se refiere a la capacidad que tiene éste para hacer que su funcionamiento sea más claro para una audiencia.

- Comprensibilidad: Se refiere a la habilidad de un algoritmo de aprendizaje a representar su conocimiento adquirido en métricas entendibles para humanos.

- Transparencia: Un modelo es transparente si por sí mismo es entendible.

2.12.1 Principales aplicaciones de la inteligencia artificial explicable

Enseguida se nombran algunas aplicaciones donde la necesidad de que la IA sea explicable es crucial, tomadas de [22].

- Cuidado médico:Los algoritmos de IA han demostrado su capacidad para sintetizar y extraer conocimiento valioso de enormes cantidades de datos médicos. En la IA médica es necesario contar con el consentimiento informado del paciente, lo que significa compartir la toma de decisiones entre médicos y pacientes de tal manera que los pacientes tengan la opinión final. Por lo tanto, la IA médica solo puede aplicarse si y solo si los pacientes son informados sobre sus funcionalidades cruciales de antemano, de manera comprensible.

- Robótica y automatización: A medida que los robots se vuelven más sofisticados e independientes, existe una creciente necesidad de que los humanos comprendan lo que hacen y piensan. La capacidad de explicar el conocimiento y la lógica detrás de las decisiones, como en las relaciones humanas, mejora sustancialmente la confianza al desarrollar una comprensión de por qué se tomó una decisión y, en consecuencia, ofrece información sobre las decisiones futuras.

- Sistema de justicia: El objetivo principal de establecer sentencias explicables es permitir que la información comprenda las conclusiones de hecho y evitar que los modelos afectan la libertad, seguridad o privacidad de las personas, debido a la discriminación u otros actos ilegales.

- Ciberseguridad: Los ataques masivos realizados por malware como virus, puertas traseras, spyware, troyanos y gusanos han presentado una amenaza de alta seguridad para los dispositivos de los usuarios, por lo que los analistas de virus necesitan clasificadores transparentes e interpretables para detectar malware de la lista gris(una lista de entidades discretas que aún no se han establecido como benignas o maliciosas).

2.13 Algoritmo LIME

LIME (siglas del inglés: *Local Interpretable Model-Agnostic Explanations*, o "Explicaciones locales e interpretables independientes de modelo") es un esquema (*framework*) de código abierto, desarrollado por Ribeiro et al. [1]. El objetivo de LIME es descubrir el proceso de toma de decisiones de un modelo de aprendizaje automático, a fin de contribuir a la certeza de su uso. La técnica intenta comprender el modelo perturbando la entrada de muestras de datos y entendiendo cómo cambian sus predicciones como consecuencia de estas predicciones.
LIME refiere en su nombre el concepto de interpretabilidad local, lo que significa que LIME se enfoca solamente en interpretar\explicar la decisión del modelo en un único

punto de datos, a diferencia de otros modelos, que son del tipo de interpretabilidad global, la cual implica interpretar el modelo como un todo sobre el conjunto de datos completo. Por otra parte, se dice que LIME es un modelo agnóstico, debido a que su técnica puede ser aplicada a cualquier modelo de aprendizaje automático de caja negra [23].

2.13.1 Representaciones de datos interpretables

Las explicaciones interpretables necesitan usar una representación que sea comprensible para los humanos, independientemente de las características reales utilizadas por el modelo. Por ejemplo, una posible representación interpretable para la clasificación de texto es un vector binario que indica la presencia o ausencia de una palabra, aunque el clasificador puede usar características más complejas (e incomprensibles). Del mismo modo, para la clasificación de imágenes, una representación interpretable puede ser un vector binario que indica la "presencia" o "ausencia" de un parche contiguo de píxeles similares (conocido como superpíxel), mientras que el clasificador puede representar la imagen como un tensor con tres canales de color por píxel. Formalizando la idea anterior, puede denotarse $x \in \mathbb{R}^d$ como la representación original de una instancia que se está explicando, y usar $x^{'} \in \{0,1\}^{d^{'}}$ para denotar el vector binario para su representación interpretable.

2.13.2 Compensación fidelidad-interpretabilidad

Se define una explicación como un modelo $g \in G$, donde G es una clase de modelos potencialmente interpretables, como los modelos lineales, arboles de decisión, por ejemplo un modelo $g \in G$. Se puede presentar fácilmente al usuario con artefactos visuales o textuales [1]. El dominio de g está definido como $\{0,1\}^{d^{'}}$; g actúa como la presencia\ausencia de componentes interpretables. Dado que no todos los $g \in G$ son

lo suficientemente simples para ser interpretados, por lo tanto se define $\omega(g)$ como una medida de la complejidad (en oposición a la interpretabilidad) de la explicación $g \in G$. Por ejemplo, para modelos lineales, $\omega(g)$ puede ser el número de pesos distintos a cero en un modelo neuronal. [1]

Denotemos el modelo que se está explicando $f : \mathbb{R}^d \to R$. En clasificación, $f(x)$ es la probabilidad (o un indicador binario) que x pertenece a una determinada clase. Además, se utiliza $\pi_x(z)$ como medida de proximidad entre una instancia z y x, para definir la localidad alrededor de x. Finalmente definimos $L(f, g, \pi_x)$ como una medida de cuán cerca coincide el modelo de explicación g con la predicción del modelo original, también conocida como fidelidad. La fidelidad local se refiere a la necesidad de que la explicación represente con precisión el comportamiento del clasificador "alrededor" de la instancia que se predice sin mirar al modelo, de ahí el enfoque independiente del modelo.

Para garantizar tanto la interpretabilidad como la fidelidad local, se debe minimizar $L(f, g, \pi_x)$ mientras tanto teniendo $\omega(g)$ lo suficientemente bajo para ser interpretado por los humanos. La explicación producida por LIME es obtenida mediante la siguiente expresión:

$$\xi(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \omega(g) \tag{4}$$

El objetivo es reducir g. Esta formulación se puede utilizar con diferentes familias de explicación G, funciones de fidelidad L y medidas de complejidad ω. Se centra en modelos lineales dispersos como explicaciones, y en realizar la búsqueda utilizando perturbaciones.

La función de pérdida L está dada por:

$$L(f, g, \pi_x) = \sum_{x' \in X'} \left[f(x' - g(x')) \right]^2 \pi_x(x') \tag{5}$$

2.13.3 LIME para imágenes

Para el caso de modelos que trabajan con imágenes, el proceso de perturbación de LIME se aplica a través de usar segmentación. En otras palabras, LIME separa la entrada a analizarse en piezas denominadas superpíxeles. Por lo tanto, genera la vecindad reemplazando arbitrariamente los superpíxeles con un color regular, probablemente imparcial. Esta vecindad se pasa más tarde a la caja negra, para resaltar los superpíxeles con peso positivo hacia una clase específica, ya que dan intuición de por qué el modelo pensaría que esa clase puede estar presente [[1], [23], [24]].
La Fig. 17 muestra un ejemplo del funcionamiento de LIME: (a) es la imagen original; (b) muestra la separación por superpíxeles en la imagen original; (c) muestra cuatro ejemplos donde la imagen original tiene porciones donde se remplazan arbitrariamente los superpíxeles con color negro; (d) muestra la explicación dada por LIME, es decir, qué superpíxeles tuvieron un peso positivo en la clase específica.

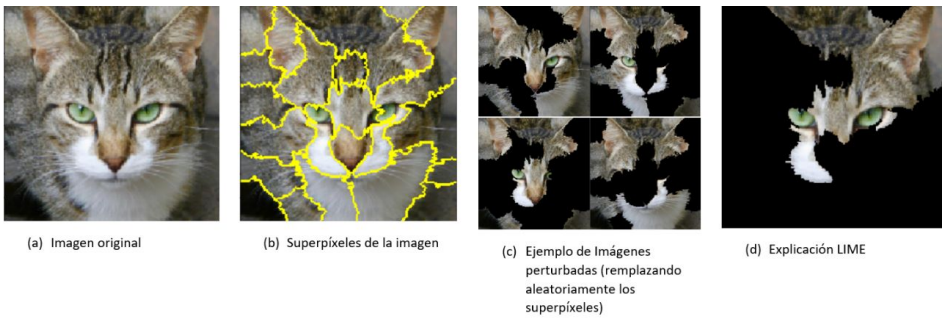


Figura 17: Un ejemplo del uso de LIME para imágenes
(ver texto para explicación)

2.14 SUS: Escalas de Usabilidad del Sistema (*System Usability Scale*)

El sistema de Escalas de Usabilidad (SUS) es un cuestionario estandarizado, ampliamente utilizado para la evaluación de la usabilidad de un sistema de software percibida por sus usuarios, desarrollado por J. Brooke [3]. En su forma estándar,

que es la más utilizada, SUS está compuesto de 10 preguntas de opción múltiple, cada una con cinco posibles respuestas, y con tono positivo y negativo alternados. La Fig.18 muestra el cuestionario completo.

	Sistema de Escalas de Usabilidad Versión estándar		Totalmente en desacuerdo				Totalmente de acuerdo
			1	2	3	4	5
1	Creo que me gustaría utilizar este sistema con frecuencia		o	o	o	o	o
2	Encontré el sistema innecesariamente complejo		o	o	o	o	o
3	Pensé que el sistema era fácil de usar		o	o	o	o	o
4	Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema		o	o	o	o	o
5	Encontré que las diversas funciones de este sistema estaban bien integradas		o	o	o	o	o
6	Pensé que había demasiada inconsistencia en este sistema		o	o	o	o	o
7	Me imagino que la mayoría de la gente aprendería a utilizar este sistema muy rápidamente		o	o	o	o	o
8	Encontré el sistema muy complicado de usar		o	o	o	o	o
9	Me sentí muy seguro usando el sistema		o	o	o	o	o
10	Necesitaba aprender muchas cosas antes de empezar con este sistema		o	o	o	o	o

Figura 18: Cuestionario estándar SUS [3]

2.14.1 Puntuación del cuestionario SUS

El cuestionario SUS produce un solo número, en un rango del 0 al 100, que representa una medida compuesta de la usabilidad general del sistema que se está estudiando. Se debe aclarar que las puntuaciones de las preguntas individuales no son significativas por sí solas, sino en conjunto es que presentan su evaluación.

El primer paso para calcular la puntuación es convertir las puntuaciones de las preguntas en puntuaciones ajustadas, (también conocidas como "contribuciones de puntuación") que van desde el 0 (calificación más baja) a 4 (mejor calificación); ese ajuste difiere para las preguntas pares e impares, que corresponden respectivamente a las preguntas de tono positivo y negativo. El sistema de puntuación del SUS requiere calificaciones para las 10 preguntas, por lo que si un encuestado deja una pregunta en blanco, se le debe dar una puntuación de 3 (el centro de la escala de

cinco puntos). Para los elementos impares, se debe restar 1 a la puntuación dada, y para los elementos pares, hay que restar 5 a la puntuación dada. Enseguida se calcula la suma de los puntajes ajustados, y se multiplican por 2.5 para obtener el puntaje estándar [25]. En la ecuación 6 se define este procedimiento:

$$\begin{aligned} \text{SUS} = 2.5(20 + \text{SUM}(\text{SUS01}, \text{SUS03}, \text{SUS05}, \text{SUS07}, \text{SUS09}) \\ -\text{SUM}(\text{SUS02}, \text{SUS04}, \text{SUS06}, \text{SUS08}, \text{SUS10})) \end{aligned} \tag{6}$$

Para representar de forma más cualitativa esta medida SUS, en [26] crearon la escala de calificación curva, la cual se muestra en la Tabla 1.

Tabla 1: Escala de calificación Curva de [26]

Rango de puntuación SUS	Calificación
84.1-100	A+
80.8-84.0	A
78.9-80.7	A-
77.2-78.8	B+
74.1-77.1	B
72.6-74.0	B-
71.1-72.5	C+
65.0-71.0	C
62.7-64.9	C-
51.7-62.6	D
0.0-51.6	F

2.15 Cálculo del tamaño de la muestra

Determinar el tamaño apropiado de la muestra de encuestados seleccionados al azar es crucial para reducir los errores de muestreo y los sesgos en las investigaciones. No

se trata simplemente de calcular la proporción en relación con la población, sino de considerar la diversidad presente en la población, los objetivos de los investigadores y las técnicas estadísticas utilizadas en última instancia. El tamaño de la muestra debe tener en cuenta todos estos factores para garantizar la representatividad y la precisión de los resultados obtenidos.

Existen diversas metodologías para calcular el tamaño de muestra en función de los objetivos y el diseño de un estudio. En la Eq. (7) se presenta una fórmula frecuentemente utilizada para calcular el tamaño de muestra en encuestas cuando se trabaja con una población finita ennumerable, es decir, una población que se puede contar [27]:

$$n = \frac{N * X}{X + N - 1} \tag{7}$$

donde:

- n es el tamaño de la muestra requerida.
- N: es el tamaño de la población universo.
- X es el valor calculado a través de la ecuación:

$$X = \frac{Z^2_{\alpha\backslash 2} * P(1-P)}{MDE^2}$$

donde a su vez:

- $Z_{\alpha\backslash 2}$ es el valor crítico de una distribución normal a $\alpha\backslash 2$. Para un nivel de confianza del 95%, α es 0.05 y el valor crítico es 1.96. Si aceptamos una $p < 0.05$ como significante, queremos decir que vamos a aceptar que la probabilidad de que el resultado se observe debido al azar, y no debido a nuestra intervención es del 5%. Para decirlo en otras palabras, estamos dispuestos a aceptar la detección de una diferencia 5 de cada 100 veces (es decir, obtener un resultado "falso positivo") [28]. La mayoría de los investigadores utilizan el nivel de confianza del 95% [29].

- MDE: Indica la cantidad de margen de error permitida.
- P: Es la proporción o porcentaje de la muestra que tiene un atributo de interés definido. En las investigaciones es común optar por la opción de máxima variabilidad, esto cuando no existen antecedentes sobre la investigación realizada. Para estos casos se establece un valor estándar de $P = 0.5$ (50%). Esto debido a que cuando el valor P es igual a 0.5 se obtiene el tamaño de la muestra más grande posible, mientras que si aumenta o disminuye el tamaño de la muestra disminuye.

2.16 Prueba t

La prueba t, también conocida como prueba t de *Student*, es una técnica de análisis estadístico desarrollada por Gosset [30] en 1908, como un medio para controlar la calidad de cervezas. Una prueba t es utilizada para probar si hay una diferencia entre dos medias de muestras independientes [31], a través de probar la hipótesis nula de que no hay diferencia entre las medias de los dos grupos [32]. La contribución de esta prueba, específicamente, es para comparar dos muestras de tamaño ≤ 30 [33].

2.16.1 Metodología de la prueba t

Para obtener resultados de la prueba t, se siguen los siguientes pasos [33]:

1. Probar que cada una de las muestras tiene una distribución normal.
2. Obtener, para cada una de las muestras:
   (a) El tamaño de las muestras: n_1 para la primera muestra y n_2 para la segunda muestra).
   (b) Sus respectivas medias: m_1 y m_2.

(c) Sus varianzas: v_1 y v_2.

3. Probar que las varianzas son homogéneas.

4. En caso de homogeneidad en las varianzas:

   (a) Establecer la diferencia entre las medias: $m_1 - m_2$.

   (b) Calcular la varianza común de las dos muestras a través de la ecuación:

$$vc = \frac{(n_1 - 1) * v_1 + (n_2 - 1) * v_2}{n_1 + n_2 - 2} \tag{8}$$

   La varianza común (vc) es igual a un promedio pesado de las varianzas de las dos muestras, en donde los pesos para ese promedio son iguales al tamaño, menos uno $(n - 1)$ para cada una de las muestras.

   (c) Se calcula el error estándar de la diferencia de las medias a través de la ecuación:

$$ESM = \sqrt{(vc) * \frac{n_1 + n_2}{n_1 * n_2}} \tag{9}$$

5. Finalmente, el valor t es igual al cociente de la diferencia de las medias entre el ESM anterior.

6. Teniendo el valor t, se procede a encontrar el valor crítico, consultando la tabla de distribución *t-Student* con grado de libertad $df = n1 + n2 - 2$. Tomando las regiones críticas de ambos lados, se rechaza la hipótesis nula si el valor t es mayor o igual que el valor crítico derecho o menor o igual que el valor crítico izquierdo. Si el valor t se encuentra fuera de la región crítica no se rechaza la hipótesis nula [34].

3 Capítulo 3: Trabajos relacionados

En este capítulo se presentan algunos trabajos que utilizan al algoritmo de IAX LIME, cuyas aplicaciones están relacionados con la del presente trabajo. Además se muestran algunos trabajos que evalúan el desempeño de LIME o lo comparan con otros modelos de explicabilidad.

3.1 Ejemplos de uso de LIME

En la literatura se encuentra una gran cantidad de aplicaciones donde LIME se utiliza para mejorar la experiencia de los usuarios en sistemas basados en aprendizaje de máquina. Considerando aquellas asociadas con el contexto de esta investigación nombramos las siguientes:

Sahay y colaboradores [35] se centraron en la tarea de asignar subtítulos explicativos de una imagen, empleando técnicas de XAI como LIME para explicar las predicciones de modelos complejos de subtítulos de imágenes. Este trabajo demuestra la aplicación de LIME en un modelo de subtítulado de imágenes, dónde representa visualmente la parte de la imagen correspondiente a una palabra, en particular en el título, justificando así el porqué el modelo predijo esa palabra. Los autores concluyeron que las explicaciones generadas por LIME para los subtítulos no son perfectas, considerando entonces que LIME no es un algoritmo completamente confiable, y puede generar explicaciones sesgadas. Los autores consideran que este sesgo se debe a la aleatoriedad en las perturbaciones que genera el modelo al ejecutarse.

Nikith y colaboradores [16] presentaron un modelo para clasificación de imágenes de flores, basado en redes neuronales convolucionales, examinando las predicciones obtenidas con LIME. Los autores concluyeron que, aunque haya modelos con un buen resultado en exactitud, estos no son explicables, por lo que mediante el uso de modelos como LIME se puede observar si el modelo esta identificando las carac-

terísticas significativas o no.

En el artículo de Sarp y colaboradores [36] se presentó un modelo que utiliza una técnica de Inteligencia Artificial Explicable (XAI) para detectar e interpretar imágenes de radiografías de tórax positivas, para el diagnóstico de la enfermedad COVID-19. En el artículo también se analiza el impacto en la decisión que se produce al utilizar mapas de calor sobre las imágenes de tórax. El modelo de solución presentado por los autores aprovecha aprendizaje de transferencia y las técnicas de aumento de datos para lograr un entrenamiento más rápido y efectivo. Además, se aplica un algoritmo para segmentación pulmonar, a fin de mejorar el rendimiento del modelo. Se realiza una comparación entre una red preentrenada y el modelo ResNet, que muestra un rendimiento de clasificación con una puntuación F1 del 98%. En este artículo se empleó el método de explicación de mapa de calor basado en LIME para detectar y encontrar COVID-19 en escaneos de radiografía de tórax. Como se ha comentado anteriormente, los modelos tradicionales de aprendizaje profundo y redes neuronales artificiales son cajas negras, lo que dificulta su comprensión y uso en el diagnóstico médico. Considerando que la XAI aborda esta limitación y proporciona herramientas que apoyan ofreciendo explicaciones que ayudan en la transparencia, este estudio propone un modelo XAI para detectar casos de COVID-19 y brindar información detallada sobre el diagnóstico. Las principales contribuciones de este trabajo son: Detección y clasificación de casos de COVID-19 a partir de imágenes de radiografía de tórax asequibles y una interpretación automática de casos de COVID-19 utilizando una implementación de mapa de calor basado en LIME con XAI a partir de radiografías para ayudar a los médicos y radiólogos. El trabajo concluyó que el mapa de calor proporcionó una mejor explicación para las áreas de neumonía donde es imposible notar la diferencia de color a simple vista. Asimismo, los experimentos realizados por los autores mostraron que estas imágenes de color, permitieron que los profesionales de salud que no tienen mucha experiencia, pudieran identificar fácilmente el área en las radiografías donde se encontraban las lesiones.

Peng y colaboradores [37] presentan un trabajo donde se centraron en resolver la in-

terpretablidad de modelos complejos, usando herramientas basadas en aprendizaje profundo, para el diagnóstico asistido por computadora en la detección de hepatitis. Los autores hacen notar que este tipo de modelos logran un buen rendimiento, debido a las características no lineales que se encuentran presentes en los datos clínicos de las aplicaciones en el mundo real. Sin embargo, como se ha comentado ya, los autores hacen notar en su artículo que estos modelos complejos de detección son considerados cajas negras, lo que genera desconfianza entre los médicos, ya que no comprenden cómo se toman las decisiones. Por lo tanto, los autores presentaron un marco de solución basado en XAI para auxiliar al profesional de la salud, en la interpretación global y local del diagnóstico de la hepatitis, sin perjudicar el buen rendimiento de la predicción ofrecido por modelos basados en aprendizaje de máquina. Asimismo, los autores muestran los resultados obtenidos con otros modelos, considerados por los usuarios como modelos transparentes, como son la regresión logística, árbol de decisión y k vecinos más cercanos, junto con modelos complejos como el algoritmo *XGBoost*, máquina de vectores de soporte y bosques aleatorios, y se aplicaron para predecir el deterioro producido por la hepatitis. Posteriormente los autores aplican técnicas de XAI, como SHAP (*SHapley Additive exPlanations*, Explicaciones Aditivas de *SHapley*), LIME y PDP (*Partial Dependence Plots*, o Gráficos de Dependencia Parcial), para mejorar la interpretación del modelo de la enfermedad hepática. Los autores reportaron que el modelo complejo de bosques aleatorios tuvo la mayor precisión entre todos los modelos analizados. Este marco de explicabilidad propuesto por los autores combina los métodos interpretables globales y locales, lo cual ayudó a mejorar la percepción de los usuarios en cuanto a la transparencia de las predicciones de los modelos complejos. Los autores concluyen que esto podría ayudar a los científicos de datos clínicos a diseñar una estructura más apropiada para el diagnóstico asistido por computadora.

Un trabajo interesante es el presentado por Zhang y colaboradores [38], donde realizaron una revisión de las tendencias recientes de aplicaciones de inteligencia artificial en el diagnóstico médico y aplicaciones quirúrgicas en conjunto con métodos de

XAI. Los trabajos analizados por los autores fueron publicados entre el 2021 y 2022 y se encontraron en los repositorios *PubMed*, *IEEE Xplore*, *Association for Computing Machinery* y *Google Scholar*. En total, los autores incluyeron 27 trabajos en diagnóstico y cirugía que utilizaron inteligencia artificial explicable. Cabe hacer notar que la aplicación del algoritmo LIME fue el enfoque XAI más utilizado en estos documentos, ya que el 25.9% (7/27) de los artículos utilizaron a LIME para explicar el modelo de aprendizaje automático propuesto. Por último, los autores concluyeron que los diferentes modelos de aprendizaje automático o modelos de aprendizaje profundo serían soluciones óptimas para diversas aplicaciones médicas apoyadas con XAI. De igual forma, los autores hacen notar que no existe algún modelo unificado de aprendizaje automático o enfoque XAI que se adapte a todas las tareas de diagnóstico y cirugía, puesto que el rendimiento de estos modelos depende del tamaño del conjunto de datos involucrado, el tipo de datos y muchos otros factores.

3.2 Evaluación de utilidad de LIME

Aún y cuando LIME tiene varios años de haberse publicado, llama la atención que no hay muchos trabajos que analicen de forma detallada su apoyo realmente efectivo hacia los usuarios en aplicaciones prácticas. Enseguida de describen brevemente los trabajos que resaltan o analizan en este enfoque.

Ding y colaboradores presentan un artículo [22] donde explican de manera general la taxonomía del área de XAI e incluyen una sección que se enfoca en los retos que presenta actualmente el estudio y uso de la XAI, incluyendo a LIME, dentro de los cuales se encuentran los siguientes:

- La XAI ha sido abordada desde diversas perspectivas en la comunidad de investigación. Sin embargo, no existe una definición estandarizada de este concepto, lo cual es esencial para establecer un marco estandarizado que guíe

a los investigadores y técnicos en el desarrollo y diseño de sistemas y soluciones en el área de XAI.

- Un reto importante que presenta la XAI es el equilibrio entre el rendimiento y la explicabilidad, por lo que se requieren más estudios para resolver este problema.

- En cuanto a las explicaciones visuales que existen actualmente, falta una forma cuantificable de evaluar la precisión e integridad del mapa explicativo que generan los modelos actuales. Este es un punto muy importante, que implica dedicar investigación a fin de poder determinar cómo expresar y explicar de la mejor manera y medir la integridad de estas explicaciones.

- Es necesario realizar más estudios para determinar métricas de evaluación en el área de la XAI, que permitan una evaluación profunda de la capacidad del algoritmo de XAI para ajustarse al significado de explicabilidad. Estas métricas deben comunicar qué tan bien se comporta un algoritmo de IA en un enfoque particular de explicabilidad.

- Las aplicaciones críticas de inteligencia artificial, como cirugía automática, vehículos autónomos y diagnóstico médico, pueden obtener resultados que amenacen la seguridad de los humanos. La existencia de este grave riesgo ha impulsado esfuerzos de supervisión exhaustivos para evitar decisiones basadas únicamente en datos. Actualmente se analiza la integración de la teoría difusa con XAI, la cual proporciona una posible solución para abordar este problema, mejorando la comprensión de la incertidumbre asociada con las decisiones críticas de IA y aumentando la confianza en sus resultados.

Dieber y Kirrane [39] examinaron la efectividad de la experiencia del usuario proporcionada por el algoritmo LIME, con un enfoque específico en su rendimiento y el objetivo de LIME de hacer que los modelos basados en datos tabulares, además

de los basados en imágenes, sean más interpretables. Para esto, los autores aplicaron varios algoritmos de aprendizaje automático de última generación en un conjunto de datos tabulares y mostraron cómo LIME se puede utilizar para complementar los métodos convencionales de evaluación del rendimiento. A partir de los resultados de experiencia del usuario, los autores evaluaron la comprensibilidad de la salida producida por LIME, tanto a través de un estudio de usabilidad que involucra a participantes que no están familiarizados con LIME, como su usabilidad general a través de un marco de evaluación personalizado, llamado Evaluación de la Usabilidad del Modelo (MUsE), el cual se deriva de la norma 9241-11: 2018 de la Organización Internacional de Normalización. En este trabajo se reportó que LIME podría mejorarse aún más a través de visualizaciones de datos autoexplicativas, con un mejor soporte para la interpretabilidad global y una mejor documentación. También se reportó que las visualizaciones proporcionadas por LIME son más adecuadas para usuarios que ya tienen experiencia trabajando con algoritmos de clasificación.

Knapič y colaboradores [40] muestran el potencial de los métodos de IAX para el apoyo a la toma de decisiones, en escenarios de análisis de imágenes médicas. Los autores usan tres tipos de métodos explicables, aplicados a un mismo conjunto de datos de imágenes médicas, con el objetivo de aumentar la comprensibilidad de las decisiones tomadas por una CNN. Los autores implementaron los métodos LIME, SHAP y el método CIU (*Contextual Importance and Utility*, Importancia y utilidad contextual) como herramientas de explicabilidad. Las explicaciones resultantes fueron evaluadas por humanos. Para esto, usuarios ajenos al área de la medicina llevaron a cabo una serie de pruebas, en un entorno de encuesta basado en la web y declararon su experiencia y comprensión de las explicaciones dadas. Los usuarios reportaron que el método explicable CIU funcionó mejor que los métodos LIME y SHAP, en términos de mejorar el apoyo a la toma de decisiones humanas y transparencia y, por lo tanto, comprensible para los usuarios. Los usuarios que recibieron soporte de explicación del método CIU respondieron un promedio de 14.90 preguntas correctamente, en comparación con los usuarios que recibieron soporte de

explicación de LIME, los cuales obtuvieron un promedio de 14.15; los usuarios que recibieron explicación de SHAP obtuvieron un promedio de 13.40. De igual forma, los usuarios que recibieron explicación de CIU mostraron una mejor compresión para distinguir entre explicaciones correctas e incorrectas. CIU obtuvo un promedio de reconocimiento de explicaciones correctas e incorrectas de 10.25, mientras que LIME y SHAP obtuvieron un promedio de 8.85 y 8.65.

En cuanto al tiempo requerido para finalizar el estudio, los usuarios de LIME mostraron una mayor rapidez con un promedio en minutos de 15.57, mientras que los usuarios de SHAP y CIU obtuvieron un promedio de 23.18 y 16.30 respectivamente. Además, CIU obtuvo un tiempo de ejecución de 8.50 segundos, lo cual superó a LIME y SHAP quienes obtuvieron tiempos de 11.40 y 9.80 segundos, respectivamente, para generar la explicación de una imagen.

Por ultimo, los autores concluyeron que existen mejoras en la implementación que pueden aplicarse en el futuro, como generalizar las explicaciones proporcionadas por los métodos explicables (LIME, SHAP y CIU), mediante el uso de diferentes conjuntos de datos médicos, lo cual podría proporcionar un mayor apoyo a la decisión para los expertos médicos. Otra sugerencia de estos autores es la de realizar un estudio de evaluación del usuario con expertos en el área médica, a fin de poder determinar con exactitud su experiencia de usuario.

Por otra parte, Del Castillo y colaboradores [41] presentaron los resultados de la comparación de LIME y CEM (*Contrastive Explanation Method*, Método de explicación contrastivo), al aplicarse sobre imágenes complejas como imágenes de expresión facial. Los autores señalan que, mientras que CEM podría utilizarse para explicar los resultados en imágenes descritas con un número reducido de características, LIME sería el método de elección cuando se trata de imágenes descritas con un gran número de características. La contribución principal de este trabajo no es en sí el modelo entrenado, sino la comprensión del cómo las explicaciones de las dos técnicas de XAI contribuyen a la comprensión humana de las predicciones generadas. Para entender el comportamiento del modelo, los autores utilizaron las herramientas de LIME y

CEM para obtener una explicación de algunas de las predicciones generadas por el modelo. El estudio concluyó que, mientras LIME genera explicaciones que pueden ser fácilmente comprensibles por los seres humanos, CEM no proporciona respuestas satisfactorias al respecto. Por lo tanto, en un escenario donde se deba elegir entre LIME y CEM, los autores proponen el uso de LIME para comprender las predicciones, en el caso de tratar con gran cantidad de características como imágenes de expresión facial, ya que la organización de la imagen en superpíxeles podría ayudar a inferir el significado a las características agrupadas. Por otro lado, CEM podría utilizarse para explicar los resultados en imágenes con un número reducido de características.

En el trabajo de Bhandari y colaboradores [42] se propone un modelo que se valida con conjuntos de datos de 7,132 imágenes de radiografía de tórax. Además, los resultados se interpretaron y explicaron utilizando LIME, el método Grad-CAM (*Gradient-weighted Class Activation Mapping* Mapeo de activación de clase ponderado por gradiente) y SHAP. Se obtuvieron los valores bien formados de SHAP, los resultados de previsibilidad de LIME y el mapa de calor de Grad-CAM, a fin de explorar el enfoque de caja negra del modelo utilizado en dicha investigación basado en aprendizaje profundo. Finalmente, con el fin de validar el modelo y calificar el riesgo médico, los autores analizaron las sensaciones médicas de clasificación, a fin de consolidar las explicaciones generadas con los modelos de XIA utilizados. Los resultados de este trabajo sugieren que los modelos de XAI y aprendizaje profundo brindan a los médicos\profesionales médicos conclusiones persuasivas y coherentes relacionadas con la detección y categorización de COVID-19, neumonía y tuberculosis.

4 Capítulo 4: Descripción del sistema

En este capítulo se detalla la estructura del modelo desarrollado para clasificar imágenes de animales usando VGG16, y el utilizado para explicar la decisión de éste a través de LIME. Con respecto al clasificador, se detallan las técnicas que se utilizaron para abordar el sobre-entrenamiento, y los pasos para llevar a cabo el entrenamiento del modelo. Asimismo, se describe la implementación de una interfaz web que, dada una imagen, permite visualizar el resultado del clasificador y la interpretación producida por LIME.

4.1 Descripción del problema de clasificación

Con el objetivo de resolver un problema por visión de computadora utilizando redes neuronales convolucionales, se planteó el problema para clasificar cinco clases distintas de animales, las cuales se representaron de la siguiente manera:

- Clase 0 como gato.
- Clase 1 como perro.
- Clase 2 como gallina.
- Clase 3 como caballo.
- Clase 4 como araña.

4.2 Conjunto de datos

Para el presente trabajo se decidió construir y utilizar para los experimentos dos conjuntos de datos distintos que se describen a continuación. Cabe mencionar que

todas las imágenes utilizadas para ambos conjuntos de datos son del formato *Joint Photographic Experts Group* (JPG).

4.2.1 Conjunto de datos con fondo irregular

Este conjunto está formado por imágenes de animales que presentan el fondo irregular, es decir, uno con una alta variabilidad de color, y que además contiene objetos que no pertenecen a ninguna de las clases involucradas en este problema. Este conjunto de datos es identificado en este trabajo como "dataset_irregular". La Fig. 19 muestra algunos ejemplos de este conjunto de datos. Las imágenes para crear "dataset_irregular" fueron recabadas en [43] y [44].

Figura 19: Ejemplos del conjunto de entrenamiento "dataset_irregular" [[43] y [44]]

El conjunto de imágenes se dividió en tres partes para utilizarse durante el entrenamiento, ajuste y validación final del clasificador, de la siguiente forma:

- Entrenamiento: 5,000 imágenes, 1,000 de cada clase.
- Validación: 2,500 imágenes, 500 de cada clase.
- Prueba: 5,000 imágenes, 1,000 de cada clase.

4.2.2 Conjunto de datos con fondo regular

Este conjunto fué creado de forma que las imágenes presentaran el fondo regular, es decir, el fondo cuenta con una menor variabilidad de color; de igual forma se trató

de evitar que las imágenes incluyeran objetos ajenos a la clase que representan. Este conjunto de datos es identificado como "dataset_regular". Se pueden observar ejemplos de este conjunto en la Fig. 20.

Figura 20: Ejemplos del conjunto de entrenamiento "dataset_regular"

Este conjunto de imágenes se dividió de la siguiente forma para la realización de los experimentos presentados en este trabajo:

- Entrenamiento: 1,000 imágenes; 200 de cada clase.
- Validación: 500 imágenes; 100 de cada clase.
- Prueba: 500 imágenes; 100 de cada clase.

Las imágenes para crear este conjunto de datos fueron recabadas en [45–58]. Dado que estas características de imágenes son difíciles de encontrar, se tuvo que recopilar imágenes de muchas fuentes, así como reducir el número total de imágenes del conjunto.

4.3 Implementación de programas de clasificación usando transferencia de aprendizaje

El modelo de clasificación fue diseñado utilizando el medio ambiente de programación *Google colaboratory*, el lenguaje Phyton y la librería *Tensorflow*, disponibles en dicho ambiente. Para esta implementación se realizaron los siguientes pasos:

1. Construcción del clasificador usando VGG16.

2. Entrenamiento de la red.

4.3.1 Estructura del modelo

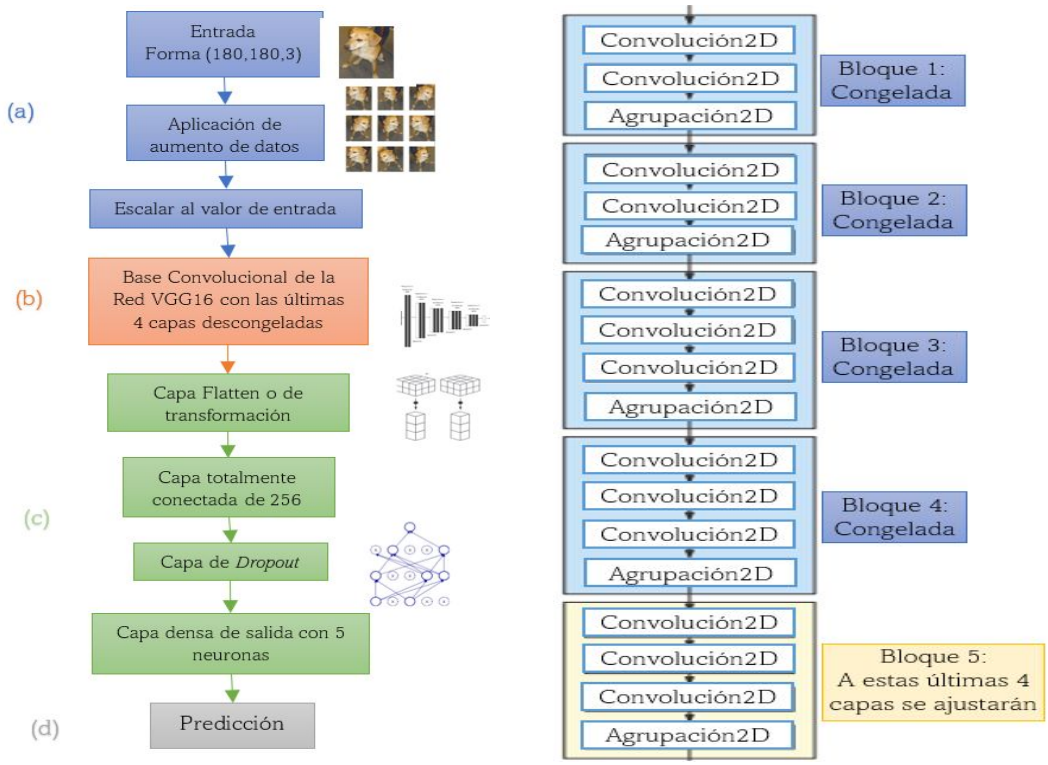
El diagrama del modelo se puede observar en la Fig. 21a; éste a su vez se puede dividir en 4 partes principales, que se describen a continuación:

- Parte a: Procesamiento de la entrada, la cual admite imágenes RGB de tamaño de $180x180$ píxeles; a las entradas se le aplica la técnica de aumento de datos. Por último, se preprocesa la imagen para que pueda ser compatible con la red VGG16: las imágenes se convierten de RGB a BGR, luego cada canal de color es centrado en cero con respecto al conjunto de datos de "ImageNet", sin escalar.

- Parte b: Se implementa la base convolucional de la red VGG16 con la técnica de ajuste fino para las últimas 4 capas de la red que están descongeladas, esto se detalla en la Fig. 21b.

- Parte c: Esta sección corresponde propiamente al clasificador. La información de la red VGG16 se transforma para adecuar la dimensión a un vector de $1xn$ con la capa *Flatten*, seguida de una capa totalmente conectada de 256 neuronas. Después se aplica la técnica *Dropout* con una probabilidad de ajuste de 0.5. Finalmente se aplica una capa totalmente conectada de 5 neuronas.

- Parte d: Salida de la última capa de 5 neuronas, la cual corresponde a la predicción del modelo.



(a) Estructura del modelo de clasificación basado en VGG16

(b) Ajuste del último bloque convolucional de la red VGG16 [13]

Figura 21: Estructura del modelo

En el código 1 (apéndice A.1) se observa la definición del aumento de datos, con los siguientes parámetros (ver sección 2.9.3): *RandomRotation* con $x = 0.1$, *RandomZoom* con $x = 0.2$ y *RandomFlip* con el eje "horizontal". Con el objetivo de generar más datos de entrenamiento a partir de los datos existentes.

La implementación de la base convolucional de la red VGG16 se observa en el código 2 (apéndice A.1) en las líneas [1-3], los parámetros utilizados para la base convolucional son:

- *weights*: Especificamos el punto de control de pesos desde el que inicializaremos el modelo; en nuestro caso escogemos el conjunto de datos *imagenet*.
- *include_top*: En nuestro caso no se incluye el clasificador predeterminado, ya

que el clasificador original cuenta con 1,000 clases del conjunto de datos *imagenet* y nuestro sistema solo requiere 5 clases.

En las líneas [5-7] del código 2 se congelan todas las capas de la base convolucional excepto las últimas 4 capas del modelo preentrenado VGG16, para poder aplicar el ajuste fino con nuestro conjunto de datos.
La construcción del modelo se observa en el código 3 (apéndice A.1). La cual se compone del procesamiento de la entrada y de la aplicación del aumento de datos (líneas [1-7]), posteriormente se implementa la base convolucional. Después se contruyó el clasificador con una capa de transformación, una capa totalmente conectada, una capa de *dropout* y una capa densa de 5 neuronas, dónde cada neurona representa una clase (líneas [8-17]). Por último, el modelo se crea a partir de las entradas y las salidas (línea 20).

4.3.2 Entrenamiento de los modelos de clasificación

Los parámetros de compilación que se utilizaron se observan en las líneas [2-4] del código 4, y son los siguientes:

- Función de pérdida: *sparse_categorical_crossentropy*.
- Optimizador: Propagación de raíz cuadrática media (RMSprop) con un factor de aprendizaje de $1e^{-5}$; este factor es muy pequeño, ya que solo se requiere hacer un pequeño ajuste en base a nuestro conjunto de datos.
- La métrica que se estará observando durante el proceso es la exactitud (*accuracy*).

Se utilizaron 10 épocas para la etapa de entrenamiento, usando el conjunto de entrenamiento y el conjunto de validación [lineas 6-10].

A partir de este momento se entrenaron dos modeles con la misma estructura y épocas de entrenamiento, pero diferente conjunto de entrenamiento y validación.

- Modelo_irregular: Este modelo fue entrenado con el conjunto de datos con fondo irregular (Fig. 19).
- Modelo_regular: Este modelo fue entrenado con el conjunto de datos con fondo regular (Fig. 20).

4.4 Implementación en *Google colaboratory* del algoritmo LIME

La siguiente implementación y descripción del algoritmo LIME fue tomada de [59]. A continuación se describe brevemente el funcionamiento del código que se muestra en el apéndice A.2:

- Líneas 1-7: Se carga el modelo de clasificación. Después se lee la imagen que se va a explicar y se redimensiona a $180x180$ píxeles. Se utiliza al modelo para predecir la clase de la imagen (variable "pred").
- Líneas 8-9: Los superpíxeles se generan utilizando el algoritmo de segmentación *quickshift* y se obtiene el número total de superpíxeles.
- Líneas 10-11: Se definen un total de 400 perturbaciones. Se generan ceros y unos aleatorios para formar una matriz con perturbaciones como filas y superpíxeles como columnas.
- Líneas 12-19: La siguiente función "perturb_image" perturba la imagen dada ("img") basada en un vector de perturbación ("perturbation") y superpíxeles predefinidos ("segments").
- Líneas 20-25: Este es el paso más costoso computacionalmente en LIME, ya que se calcula una predicción para cada imagen perturbada.

- Líneas 26-30: Utilizando la distancia del coseno, se calcula la distancia entre cada perturbación generada y la imagen que se explica.

- Líneas 31-34: Se ajusta un modelo de regresión lineal utilizando los datos de los pasos anteriores (perturbaciones, predicciones y pesos). Cada coeficiente en el modelo lineal corresponde a un superpíxel en la imagen segmentada. Estos coeficientes representan lo importante que es cada superpíxel para la predicción del modelo.

- Líneas 35-39: Se ordenan los coeficientes para averiguar cuáles son los superpíxeles que tienen coeficientes más grandes (magnitud) para la predicción dada por el modelo. Por último, se muestran los superpíxeles más importantes de la imagen después de cubrir con color negro los superpíxeles menos relevantes.

4.5 Construcción de la interfaz web sobre un servidor local

La interfaz web tiene como objetivo que los usuarios puedan interactuar con los modeles de clasificación de imágenes implementados, así como con el algoritmo de inteligencia artificial explicable LIME. De igual forma se utilizó para observar el análisis de los experimentos realizados en este proyecto.
Para llevar a cabo la construcción de la interfaz web sobre un servidor local se realizaron los pasos que se describen a continuación:

4.5.1 Paso 1: Realización de una maqueta (*mockup*) de la interfaz web

En este paso se realizó un prototipo de la interfaz web, esto con la finalidad de tener un punto inicial para su construcción. En la Fig. 22 se muestran los prototipos de las plantillas, entre las cuales se encuentran:

- Inicio.
- Sobre nosotros.
- Proyecto.
- Prueba nuestro modelo.
- Generación de la salida de LIME.

Figura 22: Maqueta de la interfaz web

4.5.2 Paso 2: Preparación del servidor local

Con el objetivo de que el servidor local sea rápido y eficiente computacionalmente hablando, se realizó la instalación para poder trabajar con la unidad de procesamiento gráfico (del inglés *graphics processing unit*, GPU). El procedimiento fue el siguiente:

1. Instalar Anaconda con Python 3.9.
2. Instalar la versión más reciente del controlador de GPU.
3. Instalar *Cuda Toolkit* 11.2.2.
4. Descargar *Cudnn* 8.1.1.

5. Copiar los archivos dentro de cada carpeta *Cudnn* y pegarlos en la carpeta *Cuda*, como se muestra a continuación.

    - Copiar \cuda\bin\cudnn*.dll y pegar en C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.2\bin.
    - Copiar \cuda\include\cudnn*.h y pegar en C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.2\include.
    - Copiar \cuda\lib\x64\cudnn*.lib y pegar C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.2\lib\x64.

6. Crear una variable de entorno con los siguientes valores:

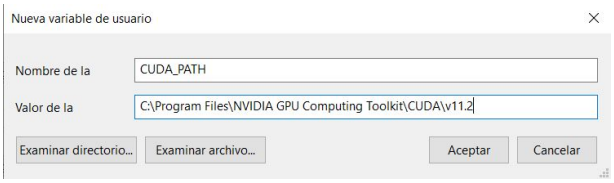


Figura 23: Variable de entorno

7. Crear un nuevo entorno *conda*.

8. Instalar *tensorflow 2.9.2* y las bibliotecas utilizadas para el proyecto, con la misma versión que *Google Colaboratory*.

Este procedimiento fue tomado de [60].

4.5.3 Paso 3: Diseño de la interfaz web

Utilizando el entorno de trabajo *flask*, *html* y *CSS* se construyó la aplicación web y se diseñaron las siguientes plantillas:

- Inicio: Está plantilla contiene la información inicial, así como la opción para probar al modelo de clasificación y el algoritmo LIME.

Figura 24: Plantilla: Inicio

- Nuestro modelo: En está sección se podrá subir una imagen y escoger el modelo de clasificación.

Figura 25: Plantilla: Nuestro Modelo

- Generador: Aquí se mostrarán los resultados del clasificador y LIME dada una imagen.

Figura 26: Plantilla: Generador

- Sobre nosotros y proyecto: Se muestra información general de los integrantes y una breve descripción del proyecto.

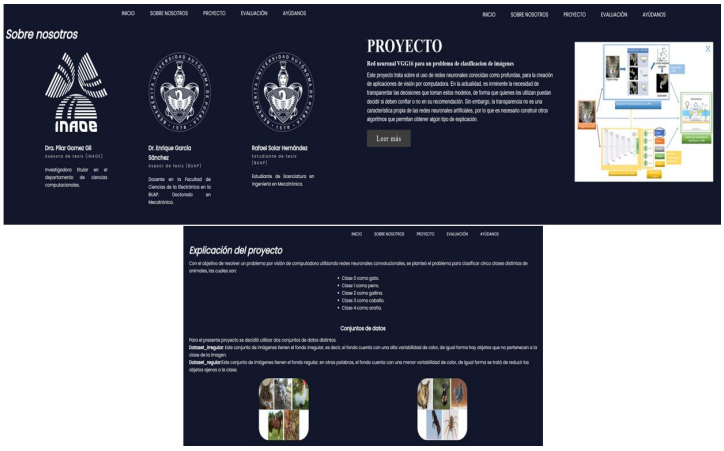


Figura 27: Plantillas: Sobres nosotros y proyecto

- Evaluación y ayúdanos: En las últimas secciones se muestran los resultados obtenidos en las evaluaciones de la interfaz y de LIME. En la plantilla de ayúdanos se muestra una encuesta para calificar al interfaz web. Cabe mencionar que estas plantillas son dinámicas, es decir, una vez que la persona usuaria responda el cuestionario SUS a través de la plantilla ayúdanos, automáticamente sus respuestas se verán reflejadas en la plantilla de evaluación de la interfaz.

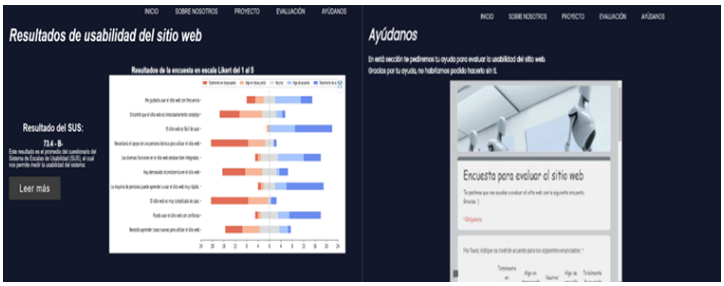


Figura 28: Plantillas: Evaluación y ayúdanos

Para poder llevar el proyecto a la práctica se realizó la conjunción de distintas herramientas, siendo las principales las siguientes:

- Modelo de clasificación de animales: En este apartado fue necesario la utilización del lenguaje de programación *Python* para la construcción del código, las bases de datos para el entrenamiento del modelo de clasificación, la biblioteca *Tensorflow 2* para desarrollar el modelo de clasificación utilizando la transferencia de aprendizaje del modelo preentrenado VGG16.

- Código del algoritmo LIME: Para llevar a cabo la implementación de este algoritmo se utilizó el lenguaje de programación *Python* y la biblioteca *scikit-image* para procesar las imágenes, así como la segmentación de las mismas.

- Interfaz web: En el este apartado se utilizó el lenguaje de programación *Python*, con la biblioteca *flask*. Por último, para crear las plantillas y su respectivo diseño fue necesario el uso de html y css.

En la Fig. 29 se muestra un diagrama de la descripción anterior.

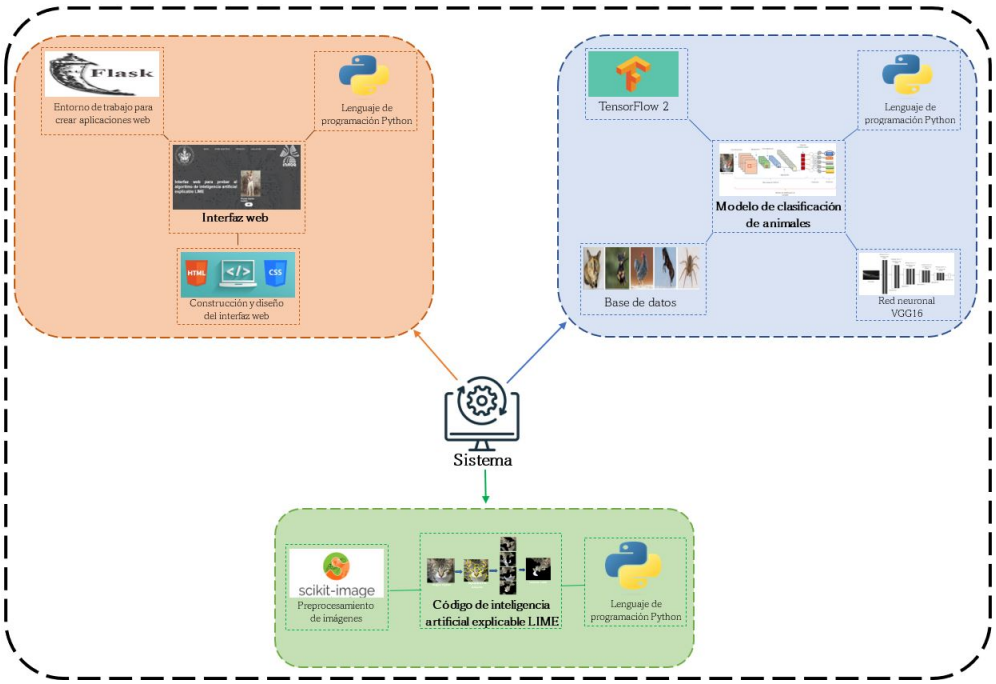


Figura 29: Diagrama de la arquitectura del sistema

5 Capítulo 5: Resultados

En este capítulo se muestran los resultados alcanzados por esta investigación, los cuales se evaluaron en sus siguientes componentes:

- Clasificaciones de la red VGG16.
- Resultados del algoritmo LIME.
- Congruencia de LIME con expectativas de usuarios comunes.
- Experiencia de los usuarios de la interfaz web.

5.1 Clasificación de la red VGG16

El desempeño de los clasificadores VGG16 se evaluó a través del uso de matrices de confusión, las cuales se describen en la sección 2.10. Se muestran enseguida las matrices obtenidas para los dos modelos de redes, entrenados a partir de las bases de datos descritas en 4.2.

- Modelo irregular. En la tabla 2 se muestra la matriz de confusión obtenida con el modelo entrenado con imágenes que presentan fondos irregulares. La diagonal de la matriz contiene el total de elementos clasificados correctamente para cada clase, es decir, los recuentos totales donde la clase real resultó igual a la clase predicha por el modelo. Por ejemplo, en el caso de la clase 0 (gato), el valor de la diagonal es 995 (primer renglón), por lo que de las 1000 imágenes del conjunto de prueba descrito en la sección (4.2.1), el modelo clasificó 995 imágenes correctamente. Los valores de cada renglón no contenidos en la diagonal, y mostrados con color anaranjado en la tabla, muestran los errores cometidos por el modelo, organizados por clase erróneamente clasificada. Por

ejemplo, el segundo valor del primer renglón (2) corresponde al número de veces que el modelo asignó la clase 1 (perro) cuando la verdadera clase es la 0 (gato); el tercer valor del primer renglón (2) es el número de veces que el modelo asignó la clase 2 (gallina) en lugar de la clase real 0 (gato) y así sucesivamente.

Teniendo la matriz de confusión definida, podemos calcular la exactitud (ver ecuación 3) del modelo, utilizando la diagonal y el número total de imágenes del conjunto de pruebas, de la siguiente forma:

$$Exactitud_{modelo\_irregular} = \frac{995 + 925 + 970 + 978 + 979}{5000} = 0.97$$

Tabla 2: Matriz de confusión del modelo "modelo_irregular", entrenado con los datos irregulares

Matriz de confusión		Predicciones del modelo				
		Clase 0	Clase 1	Clase 2	Clase 3	Clase 4
Clase real	Clase 0	995	2	2	0	1
	Clase 1	0	925	15	57	2
	Clase 2	0	10	970	14	6
	Clase 3	0	16	6	978	0
	Clase 4	1	4	6	10	979

- Modelo regular. En la tabla 3 se muestra la matriz de confusión del modelo regular, el cual está formado por 500 imágenes con fondo regular, y se describió detalladamente en 4.2.2. La exactitud obtenida en este caso es:

$$Exactitud_{modelo\_regular} = \frac{98 + 93 + 99 + 88 + 99}{500} = 0.954$$

Tabla 3: Matriz de confusión del modelo ”modelo regular”, entrenado con las imágenes depuradas

Matriz de confusión		Predicciones del modelo				
		Clase 0	Clase 1	Clase 2	Clase 3	Clase 4
Clase real	Clase 0	98	1	0	1	0
	Clase 1	2	93	1	4	0
	Clase 2	0	0	99	1	0
	Clase 3	0	9	3	88	0
	Clase 4	0	0	1	0	99

5.2 Resultados del algoritmo LIME

Utilizando la implementación del código presentado en el Apéndice A.2, se obtuvieron las salidas de la explicabilidad obtenidas por LIME, siguiendo el proceso descrito en la sección 2.13.3. En la Fig. 30 se muestran algunos ejemplos de los resultados de dicha implementación.

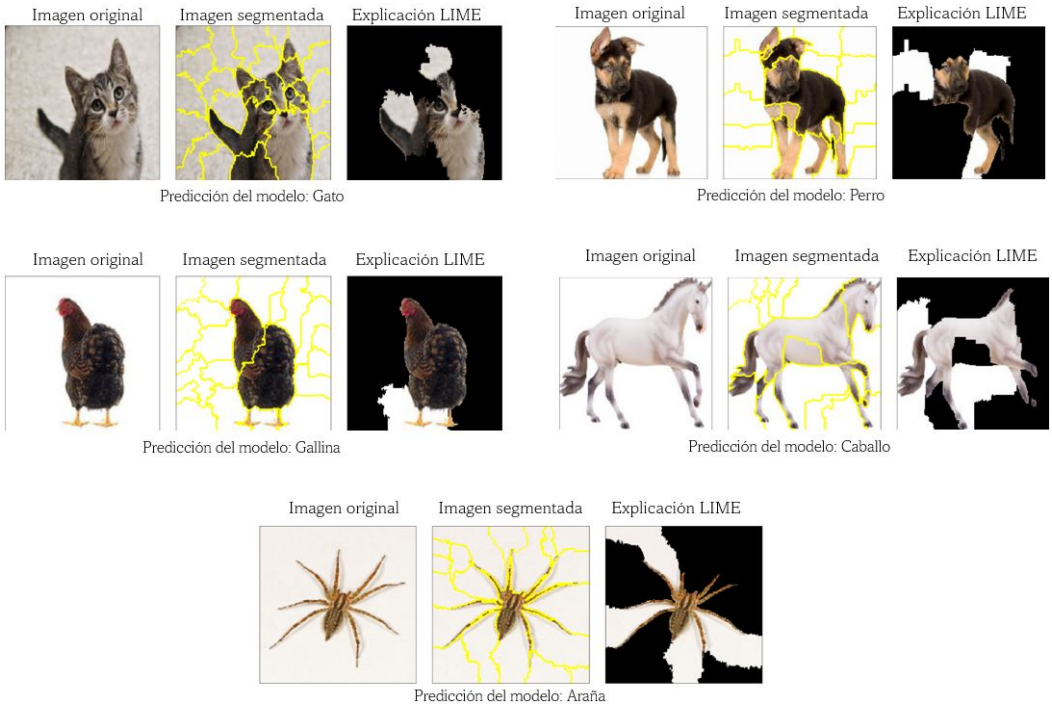


Figura 30: Ejemplos de la explicabilidad obtenida por LIME para cada clase de animales, usando el modelo regular

5.3 Congruencia de LIME con expectativas de usuarios comunes

A fin de evaluar la congruencia de LIME con respecto a la manera en que comúnmente las personas disciernen entre diferentes tipos de animales, se realizaron dos experimentos, que se describen a continuación.

5.3.1 Experimento 1: Determinación de los rasgos reconocidos como válidos por humanos, para la clasificación de animales.

Con la finalidad de determinar estos rasgos, se realizó una encuesta, que tiene como objetivo obtener los rasgos físicos que con mayor frecuencia utilizan las personas para identificar a un animal, y con ellos obtener una nube de palabras con dichos rasgos.

En el apéndice B.1 se encuentra el caso de la araña, en la figura 31e se observa la nube de palabras resultante.

Cabe aclarar que en un diagrama de nube de palabras, el tamaño de cada palabra indica la frecuencia de selección de dicha palabra, lo cual en este proyecto indica la frecuencia en que ese rasgo es marcado como útil para distinguir dicho animal. Por ejemplo, en el caso de los gatos, el rasgo que más nombraron las personas como útil para distinguirlo de otros animales es "bigotes", y en el caso de arañas lo que más usan son "patas".

La encuesta fue desarrollada y aplicada utilizando la herramienta formularios de *Google*. Para evitar que una sola persona pudiera responder la encuesta varias veces, ésta se limitó a una respuesta por correo electrónico.

Evaluación del experimento 1. Esta encuesta fue respondida 165, esto debido a la disponibilidad limitada del tiempo con el que se contó para la aplicación de la encuesta, así como al alcance de población disponible. Entonces, se realizó un análisis estadístico para determinar si este número es suficiente para realizar una encuesta válida. Cómo se explica en la sección 2.15, el tamaño de muestra válido depende de los parámetros:

Utilizando los valores:

- Tomamos un nivel de confianza de $Z_{\alpha\backslash 2} = 95\%$. Por lo tanto, el valor crítico de la distribución normal a $\alpha\backslash 2$ es $1,96$.
- $MDE = 8\%$.
- Dado que no se encontraron antecedentes sobre esta investigación y optando por obtener la máxima variabilidad. La proporción de la población se definió como $P = 50\%$ (0.5).
- Para esta encuesta se tomo a la comunidad de alumnos de la Facultad de Ciencias de la Electrónica (FCE) de la Benemérita Universidad Autónoma de

Puebla (BUAP) como la población universo. Consultando en [61] el padrón de alumnos es de 3460. De esta forma se define $N = 3460$.

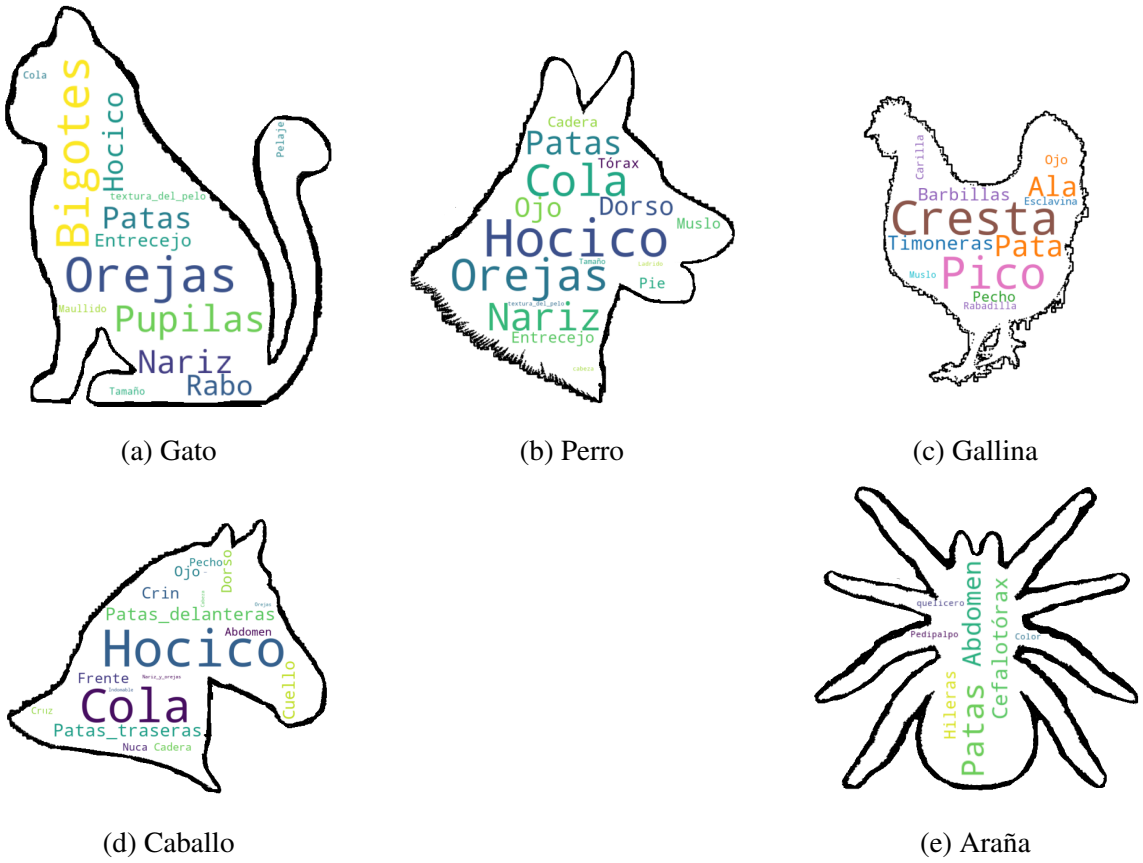
Se obtiene que el tamaño de muestra n está dado por:

$$X = \frac{1.96^2 * 0.5(1 - 0.5)}{0.08^2} = 150.063$$
$$n = \frac{3460 * 150.0625}{150.0625 + 3459} = 143.865 \approx 144$$

Donde $n < 165$ el tamaño de la muestra obtenida. La tabla 4 muestra en orden descendente, los cinco rasgos más seleccionados por las personas que respondieron la encuesta, para identificar a cada uno de los animales utilizados en este trabajo. Utilizando estos valores, y a fin de presentar los rasgos de una forma atractiva a los usuarios, se implementó un programa para generar nubes de palabras, obteniéndose el resultado que se muestra en la Fig. 31.

Tabla 4: Resultados cuantitativos de la encuesta de rasgos físicos de los animales

Gato		Perro		Gallina		Caballo		Araña	
Rasgo	Conteo	Rasgo	Conteo	Rasgo	Conteo	Rasgo	Conteo	Rasgo	Conteo
Orejas	133	Hocico	132	Cresta	157	Hocico	131	Patas	161
Bigotes	116	Orejas	117	Pico	138	Cola	109	Abdomen	103
Pupilas	107	Nariz	113	Pata	97	Patas traseras	98	Cefalotórax	71
Nariz	82	Cola	112	Ala	85	Patas delanteras	97	Quelíceros	50
Patas	80	Patas	98	Timoneras	70	Cuello	77	Hileras	34



(a) Gato (b) Perro (c) Gallina

(d) Caballo (e) Araña

Figura 31: Nube de palabras de los animales

5.3.2 Experimento 2: Evaluación del algoritmo LIME por usuarios

El objetivo de este experimento es determinar la similitud entre las explicaciones otorgadas por LIME y lo esperado por usuarios. La similitud es medida en este experimento como la coincidencia en el número de rasgos escogidos entre el sistema XIA LIME y el número de rasgos escogidos por un conjunto de personas. A fin de determinar el número promedio escogido por personas, se realizaron dos encuestas, (que denominamos Encuesta A y Encuesta B, en el apéndice). Cada encuesta contiene 20 preguntas, presentando 4 imágenes por cada clase de animal, que fueron correctamente clasificadas por VGG16. Como se observa en la Fig. 32, cada pregunta está compuesta por tres partes: 1) una imagen, donde se presentan las regiones que

seleccionó LIME para determinar el porque el sistema de clasificación asignó dicha imagen; 2) la nube de palabras que representa los rasgos físicos para distinguir el animal escogido; 3) seis posibles opciones de respuesta a la encuesta, donde se pidió que la persona respondiera si la región o regiones que seleccionó LIME:

- Corresponden a 1 o 2 palabras más frecuentes.
- Corresponden a 3 o 4 palabras más frecuentes.
- Corresponden a las 5 palabras más frecuentes.
- Al menos alguna palabra de las que aparecen en la nube.
- No corresponden a nada de lo que aparece en la nube.
- Otra, en la cual se podía escribir la razón para haber elegido esta opción.

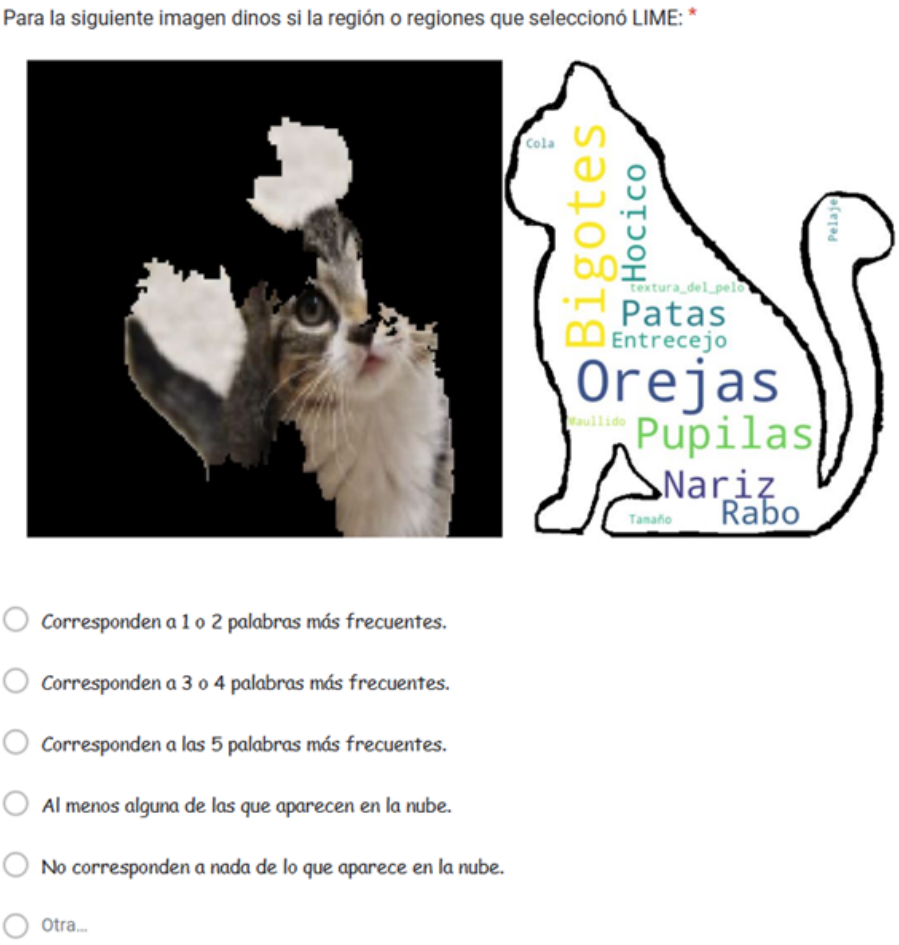


Figura 32: Ejemplo de la pregunta de la clase gato del experimento 2

La encuesta A fue respondida por 80 personas y la encuesta B por 75 personas, lo cual da un total de 155 personas. Nótese que este número es mayor que 144, el resultado de tamaño de muestra representativo de la población, descrito anteriormente en la evaluación del experimento 1. Las encuesta fueron desarrolladas con la herramienta formularios de *Google*. Para evitar que una sola persona pudiera responder varias veces la encuesta, se limitó a una respuesta por correo electrónico.

Evaluación del experimento 2:

Los resultados de las encuestas se procesaron de la siguiente manera:

1. Se calculó el número total de respuestas por cada opción y por cada imagen.

En la Tabla 5 se muestra un ejemplo, correspondiente a la imagen identificada con el número uno.

2. Para obtener los resultados totales de congruencia en cuanto al número de rasgos seleccionados, para cada posible respuesta se realizó la suma total de respuestas otorgadas a las 40 imágenes, y se calculó el porcentaje de selección de cada respuesta para las 40 imágenes, los cuales se muestran en la tabla 6.

Nótese que, de acuerdo a la tabla 6, la opción seleccionada mas frecuente es "corresponden a 1 o 2 palabras frecuentes" (38%), seguida de la opción "corresponden a 3 o 4 palabras frecuentes" (26%), dando un total de (64%). Esto implica que la mayoría de las personas escogieron las dos opciones mas bajas en cuanto a la congruencia entre lo que dice LIME y lo que esperan las personas.

Tabla 5: Ejemplo del conteo de las respuestas de personas para la imagen No. 1

Respuesta posible	Número de personas
Corresponden a 1 o 2 palabras más frecuentes	10
Corresponden a 3 o 4 palabras más frecuentes	42
Corresponden a las 5 palabras más frecuentes	27
Al menos alguna de las que aparecen en la nube	0
No corresponden a nada de lo que aparece en la nube	1

Tabla 6: Porcentaje de selección de respuestas de las encuestas A y B

Respuesta posible	Total de respuestas	Porcentaje (%)
Corresponden a 1 o 2 palabras más frecuentes	1163	38
Corresponden a 3 o 4 palabras más frecuentes	808	26
Corresponden a las 5 palabras más frecuentes	480	15
Al menos alguna de las que aparecen en la nube	248	8
No corresponden a nada de lo que aparece en la nube	401	13
Total	3100	100

5.4 Usabilidad de la Interfaz Web

Para calcular la usabilidad de la interfaz se utilizó el cuestionario SUS (*System Usability Scale*) el cual se explica con detalle en la Sección 2.14.

Se le pidió a 20 personas que interactuaran con todos los elementos de la interfaz web, incluyendo la parte en la cual la persona pudo documentarse acerca del proyecto y de sus participantes. Asimismo, estos usuarios tuvieron la oportunidad de probar los clasificadores de imágenes y al algoritmo LIME, introduciendo en la interfaz alguna imagen propia. Una vez realizada esta etapa de interacción, los usuarios debían responder el cuestionario SUS.

La Fig. 33 muestra una gráfica *Likert* de las respuestas obtenidas del cuestionario aplicado. La escala *Likert*, está diseñada para medir fácilmente las actitudes, opiniones o percepciones de las personas, done los sujetos eligen entre una gama de posibles respuestas a una pregunta o declaración específica [62]. En el cuestionario aplicado, las respuestas posibles son: "totalmente de acuerdo", "de acuerdo", "neutral", "en desacuerdo" y "totalmente en desacuerdo", por lo que la gráfica incluye 5 posibles valores para cada respuesta.

El promedio de todas las evaluaciones de SUS, calculadas según la Ecuación 6, es de 73.4. Este valor, de acuerdo a la tabla 1, corresponde a una calificación cualitativa de usabilidad de la interfaz de $B-$.

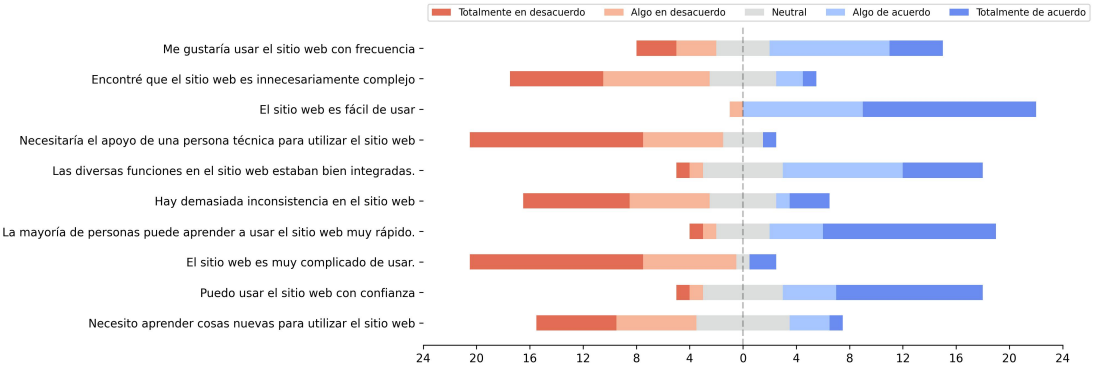


Figura 33: Gráfica likert de la interfaz web

Evaluación de significancia estadística usando la prueba t

La encuesta de usabilidad fue aplicada a dos grupos de muestras diferentes: el primer grupo estuvo formado por 13 alumnos de la Facultad de Ciencias de la Electrónica (FCE) de la Benemérita Universidad Autónoma de Puebla (BUAP) y el segundo por 7 personas ajenas a esta comunidad. Para poder determinar si los resultados obtenidos por esta muestra pueden extrapolarse a la población en general, se evaluó si hay diferencia significativa en cuanto a la media de las respuestas de las dos muestras. Teniendo en cuenta que el tamaño de las muestras recolectadas es menor a 30, se utilizó la prueba estadística t. Para esto se tomaron al azar siete respuestas del cuestionario SUS de la FCE y las siete respuestas de personas que no pertenecían a la FCE. Se escogieron siete respuestas debido a que éste es el tamaño del grupo menor entre los dos disponibles. La tabla 7 muestra los resultados SUS en el subconjunto de muestras representando la comunidad que pertenece a la FCE y el de la comunidad que no pertenece a la FCE. Cada renglón representa el valor del cuestionario SUS de un individuo.

Tabla 7: Resultados del cuestionario SUS en el subconjunto de muestras representando la comunidad que pertenece a la FCE y la comunidad que no pertenece a la FCE.

Individuos	muestra FCE	muestra no FCE
1	40.0	25.0
2	65.0	62.5
3	65.0	62.5
4	80.0	65.0
5	80.0	82.5
6	85.0	90.0
7	92.5	97.5

La hipótesis nula establece que no hay diferencias en la media de las dos muestras independientes y la hipótesis alterna establece que de existir esta diferencia, sólo se debe al azar, esto es: [33].

- Hipótesis nula: $H_0 : \mu_1 = \mu_2$.
- Hipótesis alterna: $H_1 : \mu_1 \neq \mu_2$

Para realizar esta prueba se siguió la metodología descrita en 2.16. Para encontrar los valores requeridos en dicha metodología, se siguieron los siguientes pasos:

1. Para probar la normalidad de las muestras, se utilizó la prueba de bondad de ajuste de *Shapiro-Wilks*, que permite verificar si una muestra de tamaño igual o inferior a 50 sigue una distribución normal y, por tanto, qué pruebas (paramétricas o no) podemos llevar a cabo en los análisis estadísticos [63]. Para esto se utilizó el software *SPSS* (*Statistical Package for the Social Sciences*) y la metodología descrita en [63].

   La hipótesis nula para prueba de normalidad, (H_0) establece que la muestra procede de una distribución normal, mientras que la hipótesis alternativa (H_a) es que los datos no se distribuyen según un modelo de probabilidad normal.

Por tanto, para aceptar H_0 el valor de la significancia estadística (valor de p) deberá ser mayor de 0.05 (tomando un nivel de significancia de $\alpha = 0.05$) [63]. Con el software *SPSS* se obtuvieron los valores de la prueba de *Shapiro-Wilks* que se muestran en la Fig. 34.

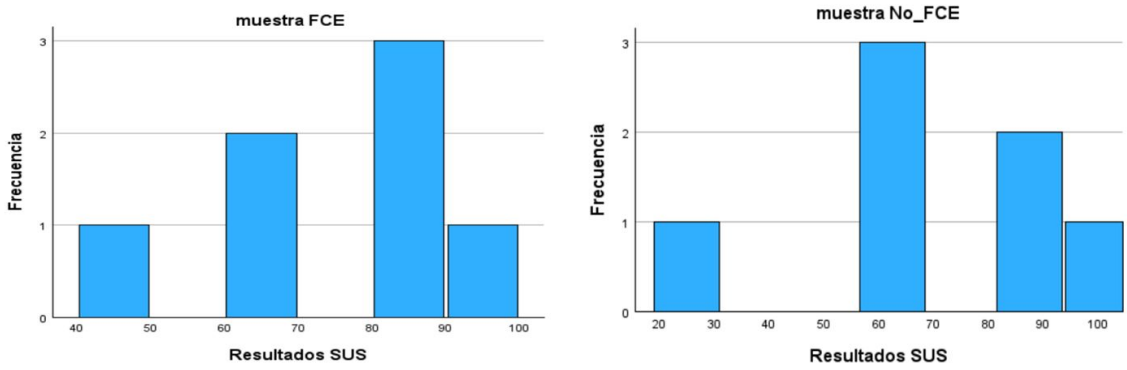
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
FCE	,237	7	,200*	,910	7	,395
No_FCE	,246	7	,200*	,914	7	,425

Figura 34: Pruebas de normalidad obtenidas con el software *SPSS*

Analizando los valores en la prueba *Shapiro-Wilks* se tiene que:

- Para la muestra FCE se obtuvo significancia estadística ($p = 0.395$). Como el valor p es mayor que 0.05, entonces se acepta H_0, y se afirma que la muestra FCE sigue una distribución normal.
- Para la muestra no FCE obtenemos la significancia estadística ($p = 0.425$). Como el valor p es mayor que 0.05, entonces se acepta H_0, y se afirma que la muestra no FCE sigue una distribución normal.

La Fig. 35a muestra los histogramas de frecuencia para la muestra FCE, y la Fig. 35b los resultados para la muestra no FCE. En el eje x se encuentran los resultados del cuestionario SUS y en el eje y la frecuencia de individuos.



(a) Histograma de la muestra FCE (b) Histograma de la muestra no FCE

Figura 35: Histograma de frecuencias de las muestras FCE y no FCE

2. En la tabla 8 se muestran los siguientes resultados obtenidos por *SPSS*: tamaño de las muestras sus respectivas medias y sus varianzas.

Tabla 8: Tamaños de muestra, medias de las muestras y sus respectivas varianzas obtenidos con el software *SPSS*.

	muestra de FCE	muestra No FCE
Tamaño (n)	7	7
Media ($\bar{X}$)	72.5	69.29
Varianza (v)	306.25	578.57

3. Para demostrar la homogeneidad de las varianzas se realizó la prueba de *Levene*, la cual indica si se puede o no suponer varianzas iguales. La prueba de *Levene* debe arrojar una significación mayor de 0.05 para que se cumpla el requisito de homogeneidad [64].

   Siguiendo la metodología propuesta en [64] y haciendo uso del software *SPSS*, se realizó la prueba *Levene* para las muestras FCE y no FCE (tabla 7), con la cual se generó la tabla que se encuentra en el apéndice C.1. Dado que es un estudio de las medias se utiliza el primer renglón de dicha tabla, esto es, $p = 0.536$ el cual es mayor es mayor a 0.05. Por lo tanto, podemos concluir que las varianzas son homogéneas.

4. Dado que las varianzas son homogéneas se procede a calcular lo siguiente:

   - Diferencia de medias: $72.5 - 69.29 = 3.21$.
   - Varianza común de las dos muestras (Ecuación 8 detallada en 2.16): $vc = \frac{(7-1)*306.25+(7-1)*578.57}{7+7-2} = 442.41$
   - Error estándar de las diferencias de las medias (Ecuación 9, detallada en 2.16): $ESM = \sqrt{(442.41) * \frac{7+7}{7*7}} = 11.24$

5. El valor de t: $t = \frac{3.21}{11.24} = 0.285$

6. Para encontrar el valor crítico con los grados de libertad $df = 7 + 7 - 2 = 12$, se consulta la tabla de valores críticos de la distribución t tomada de [34] (Apéndice C.2). Con un nivel de significancia de $\alpha = 0.05$ y grados de libertad $df = 12$ se obtiene el valor crítico de ± 1.782. Nuestro valor t es menor al valor crítico por el lado derecho ($0.285 < 1.782$) y mayor al valor crítico por la izquierda ($0.285 > -1.782$). Por lo tanto, se acepta la hipótesis nula, no hay diferencia entre los resultados del cuestionario SUS de la comunidad FCE y la comunidad ajena a la FCE.

6 Capítulo 6: Conclusiones y trabajo futuro

En esta investigación, se realizó un análisis detallado del comportamiento del algoritmo de XIA LIME, desde el punto de vista del usuario común, al aplicarse sobre un clasificador basado en el modelo de aprendizaje profundo VGG16. Además, se llevó a cabo la construcción de una interfaz web que, dada una imagen, permite visualizar el resultado del clasificador y de la interpretación producida por LIME.

De los resultados obtenidos, se concluye que a pesar de que el algoritmo de XIA LIME ayuda a sus usuarios a entender las razones de selección de la clase del clasificador, LIME no lo hace a la perfección. Se esperaría que para calificar a LIME como una excelente herramienta de explicabilidad, se presentara más congruencia de la que se obtuvo, entre las regiones que selecciona LIME y las regiones que las personas utilizan para identificar a un animal. Se puede observar en la tabla 6, que los porcentajes obtenidos indican que la mayoría de respuestas pertenecen a la opción "corresponden a 1 o 2 palabras más frecuentes"; sin embargo se hubiera esperado que el mayor porcentaje estuviera entre las opciones "corresponden a las 5 palabras más frecuentes". En otras palabras, si las personas seleccionan la opción "corresponden a las 5 palabras más frecuentes" entonces las regiones que selecciona LIME deberían corresponder prácticamente todo el animal. La Fig. 36a muestra uno de los pocos ejemplos obtenidos donde sucede este caso. Cuando las personas seleccionaron la opción "corresponden a 1 o 2 palabras más frecuentes" se debió a que las regiones que seleccionó LIME fueron solo muy pocos rasgos. La Fig. 36b muestra un ejemplo de este caso.

(a) Imagen 23 (b) Imagen 2

Figura 36: Resultados individuales

De igual forma, a lo largo de los experimentos se percibió que para usuarios comunes (con muy poca o nula experiencia en el área de modelos de clasificación) la implementación del algoritmo LIME como herramienta de explicabilidad no es muy intuitiva por sí misma. Al mostrarles a los usuarios solo la salida de LIME sin ninguna información previa acerca del funcionamiento de LIME, como se muestra en el ejemplo de la Fig. 37, resulta difícil para éstos el darle un sentido a los resultados. Estos hallazgos concuerda con los reportados en el estudio [39].

Figura 37: Ejemplo de salida LIME sin información previa

Con respecto al apartado de evaluación de la interfaz web, donde se obtuvo una calificación SUS de 73.4, concluimos que este resultado es aceptable. Se hace notar que para la mayoría de las personas la interfaz web fue fácil de usar; de igual manera se aprecia que la mayoría de personas pudo aprender a utilizar el sitio web rápido. Nótese que, debido a que la interfaz es una herramienta para trabajos específicos

acerca de explicabilidad, solo un poco mas de la mitad de usuarios contestaron que usarían el sitio web con frecuencia, mientras que los demás son neutrales o están en desacuerdo a utilizar el sitio web con frecuencia. Por su naturaleza, ésta no es una herramienta que las personas requieran usar diariamente o cada semana. Por otra parte, haciendo uso de la prueba t se determinó que no hay diferencia entre los resultados de usabilidad en el subconjunto de muestras representando la comunidad que pertenece a la FCE y la comunidad que no pertenece a la FCE.
La construcción de una interfaz web como programador fue un desafío gratificante y enriquecedor, ya que se aplicaron conocimientos y habilidades en programación específicas para el desarrollo de una interfaz web funcional. Durante el proceso de construcción, se aprendió a utilizar diversas herramientas, como HTML, CSS y *Flask*, a fin de poder brindar a los usuarios una interfaz agradable y fácil de utilizar. A lo largo de este proyecto, hubo bastantes desafíos técnicos que me permitieron mejorar mis habilidades en la implementación de sitios web e interfaces. Estos desafíos me han ayudado a mejorar mi destreza como programador y a aprender nuevas técnicas de desarrollo web.

El área de la mecatrónica, en conjunto con la visión por computadora y la inteligencia artificial explicable, ofrecen la posibilidad de desarrollar sistemas inteligentes que puedan percibir, comprender y explicar el entorno visual de manera comprensible para los seres humanos. La conjunción de estas disciplinas nos permite crear aplicaciones en diversos campos, como la robótica, la automatización industrial, vehículos autónomos y la medicina, donde se requieren sistemas autónomos y confiables que puedan generar confianza en sus usuarios con respecto a su toma de decisiones. En todas estas aplicaciones la visión computacional inteligente y los elementos de explicabilidad juegan un papel preponderante.

6.1 Trabajo futuro

Aún se requiere más investigación y estudio en el campo de la explicabilidad de los modelos de clasificación, incluyendo el algoritmo LIME, con la finalidad de aumentar la confianza de los usuarios al utilizar las recomendaciones de dichos modelos. Con respecto a la línea de IAX utilizada en este trabajo, el siguiente paso podría ser el realizar un estudio analítico para encontrar algoritmos de explicación que, desde la perspectiva de los usuarios comunes, presenten mejores resultados que los ofrecidos por LIME actualmente. De igual forma es importante analizar y comprender mejor los parámetros de LIME, a fin de mejorar la calidad de las explicaciones proporcionadas por este algoritmo. Por último, un trabajo muy importante podría ser la mejora en las explicaciones de LIME, de tal forma que LIME fuera más intuitivo para usuarios con muy poca o nula experiencia en los modelos de clasificación.
Como posible mejora en el aspecto del diseño, se podría implementar un diseño adaptable, conocido como *responsive* en la interfaz web,el cual permitiría que la interfaz se adapte automáticamente a diferentes tamaños de pantalla. En cuanto a la funcionalidad, una actualización interesante sería permitir que el usuario pueda cargar su propio modelo de clasificación de imágenes y utilizarlo junto con el algoritmo LIME para realizar pruebas de explicabilidad.

Bibliografía

[1] M. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," *In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining,*, 2016.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, vol. 6, 2014.

[3] J. Brooke, "SUS: A quick and dirty usability scale," *Usability evaluation in industry*, pp. 189–194, 1996.

[4] O. para la Cooperación y el Desarrollo Económicos, "The second international conference on ai in work, innovation, productivity and skills." https://oecd-events.org/2022-ai-wips. Consultado: 25 de Octubre 2022.

[5] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, *Computational Intelligence, a Methodological Introduction*. Springer, 2 ed., 2016.

[6] S. Russell and S. Russell, *Artificial Intelligence: A Modern Approach*. Pearson Education, global ed., 2016.

[7] I. N. da Silva et al., *Artificial Neuronal Networks. A Practical Course*. Springer, 2017.

[8] I. Livshin, *Artificial Neuronal Networks with Java. Tools for Building Neuronal Network Applications*. Apress, 2019.

[9] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[10] P. Gómez-Gil, "Introducción a la inteligencia computacional, notas de curso. Coordinación de Ciencias Computacionales, INAOE." https://ccc.inaoep.mx/~pgomez/cursos/IC-I/acetatos/introduccion.pdf, 2021.

[11] S. S. Haykin, *Neural networks and learning machines*. Upper Saddle River, NJ: Pearson Education, third ed., 2009.

[12] A. Gulli, A. Kapoot, and S. Pal, *Deep Learning with Tensorflow 2 and Keras*. Packt, 2019.

[13] F. Chollet, *Python Deep Learning*. Manning, 2 ed., 2021.

[14] D. E. R., *Computer Vision : Principles, Algorithms, Applications, Learning.*, vol. Fifth edition. Academic Press, 2018.

[15] R. Shashidhar, "Detection of choroidal neovascularization (CNV) in retina OCT images using VGG16 and densenet CNN," *Wireless Personal Communications*, pp. 2569–2583, 2022.

[16] B. V. Nikith, M. T. Nikhil, M. S. S. Siddhartha, and M. K, "LIME Explainability on Flower Classification," *2022 6th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 2022.

[17] X. Ying, "An Overview of Overfitting and its Solutions," *Conference Series*, 2019.

[18] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Deep Learning with Python*. Packt, 2 ed., 2019.

[19] D. Sarkar, R. Bali, and T. Sharma, *Practical Machine Learning with Python*. Apress, 2018.

[20] A. Barredo, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward resposible AI," *Information Fusion*, pp. 1–30, 2020.

[21] "Opening the "black box" of artificial intelligence." https://ec.europa.eu/research-and-innovation/en/horizon-magazine/

opening-black-box-artificial-intelligence. Consultado: 05 de agosto de 2022.

[22] W. Ding, M. Abdel-Basset, H. Hawash, and H. Hawash, "Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey," *Information Sciences*, p. 238–292, 2022.

[23] S. Sahay, N. Omare, and K.K.Shukla, "An Approach to identify Captioning Keywords in an Image using LIME," *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 648–651, 2021.

[24] M. S. Islam, I. Hussain, M. M. Rahman, S. J. Park, and M. A. Hossain, "Explainable Artificial Intelligence Model for Stroke Prediction Using EEG Signal," *Sensors*, 2022.

[25] J. R. Lewis, "The System Usability Scale: Past, Present, and Future," *International Journal of Human-Computer Interaction*, pp. 577–590, 2018.

[26] J. Sauro and J. R. Lewis, *Quantifying the User Experience : Practical Statistics for User Research*. Morgan Kaufmann Publishers, 2 ed., 2016.

[27] I. Etikan and O. Babatope, "A basic approach in sampling methodology and sample size calculation," *MedLife Clinics*, vol. 1, pp. 50–54, 2019.

[28] P. Kadam and S. Bhalerao, "Sample size calculation," *International Journal of Ayurveda Research*, vol. 1, pp. 55–57, 2010.

[29] S. S. Haykin, "Calculadora de tamaño de muestra." https://www.questionpro.com/es/calculadora-de-muestra.html, 2021. Consultado: 10 de junio de 2023.

[30] Student, "The probable error of a mean," *Biometrika*, vol. 6, pp. 1–25, 1908.

[31] T. K. Kim, "T test as a parametric statistic," *Korean Journal of Anesthesiology*, pp. 540–546, 2015.

[32] Z. Ali and S. B. Bhaskar, "Basic statistical tools in research and data analysis," *Indian Journal of Anaesthesia*, pp. 54–61, 2016.

[33] R. A. Sánchez, "t-student. usos y abusos," *Revista Mexicana de Cardiología*, pp. 59–61, 2015.

[34] G. K. Kanji, *100 statistical tests*. SAGE Publications, 3 ed., 2006.

[35] S. Sahay, N. Omare, and K. K. Shukla, "An approach to identify captioning keywords in an image using lime," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 648–651, 2021.

[36] S. Sarp, F. O. Catak, M. Kuzlu, U. Cali, H. Kusetogullari, Y. Zhao, G. Ates, and O. Guler, "An XAI approach for COVID-19 detection using transfer learning with X-ray images," *CellPress*, 2023.

[37] J. Peng, K. Zou, M. Zhou, Y. Teng, X. Zhu, F. Zhang, and J. Xu, "An Explainable Artificial Intelligence Framework for the Deterioration Risk Prediction of Hepatitis Patients," *Journal of Medical Systems*, 2021.

[38] Y. Zhang, Y. Weng, and J. Lund, "Applications of Explainable Artificial Intelligence in Diagnosis and Surgery," *diagnostics*, 2022.

[39] J. Dieber and S. Kirrane, "A novel model usability evaluation framework (MUsE) for explainable artificial intelligence," *Information Fusion*, pp. 143–153, 2022.

[40] S. Knapič, A. Malhi, R. Saluja, and K. Främling, "Explainable Artificial Intelligence for Human Decision Support System in the Medical Domain," *Mach. Learn. Knowl*, pp. 740–770, 2021.

[41] G. del Castillo Torres, M. F. Roig-Maimó, M. Mascaró-Oliver, E. Amengual-Alcover, and R. Mas-Sansó, "Understanding How CNNs Recognize Facial Expressions: A Case Study with LIME and CEM," *Sensors*, 2022.

[42] M. Bhandari, T. B. Shah, B. Siku, and A. Neupane, "Explanatory classification of CXR images into COVID-19, Pneumonia and Tuberculosis using deep learning and XAI," *Computers in Biology and Medicine*, 2022.

[43] A. Corrado, "Animals-10. [Dataset] Kaggle.com." https://www.kaggle.com/datasets/alessiocorrado99/animals10, 2020.

[44] Larxel, "Animal faces. [Dataset] Kaggle.com." https://www.kaggle.com/datasets/andrewmvd/animal-faces, 2020.

[45] S. Chaudhari, "Spider. [Dataset]. Kaggle.com.." \https://www.kaggle.com/datasets/sahilzngr/spider, 2019.

[46] M. Ribeiro, "Spider-scorpion-cockroach. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/mathewribeiro/spiderscorpioncochroach, 2020.

[47] Pınar, "spider species. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/efriea/spider-species, 2020.

[48] M. Waquar, "Classification-of-species(9). [Dataset]. Kaggle.com.." https://www.kaggle.com/dsv/4039276, 2022.

[49] Gerry, "Yikes! spiders -15 species classification. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/gpiosenka/yikes-spiders-15-species, 2020.

[50] O. Belitskaya, "Horse breeds. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/olgabelitskaya/horse-breeds, 2020.

[51] A. Rougetet, "horse2zebra. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/arnaud58/horse2zebra, 2019.

[52] S. Saxena, "Animal -5 mammal. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/shiv28/animal-5-mammal, 2021.

[53] A. Wahee, "dogs-cats-horses-humans-dataset.[Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/eabdul/dogs-cats-horses-humans-dataset, 2018.

[54] A. Ghzawi, "Chicken breeds. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/abdalnassir/chicken-breeds, 2021.

[55] G. Cox, "chicken-photos. [Dataset]. Github.com.." https://github.com/gunthercox/chicken-photos, 2017.

[56] S. S. Schubert, "Cat and dog. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/tongpython/cat-and-dog, 2017.

[57] J. Li, "Stanford dogs dataset. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/jessicali9530/stanford-dogs-dataset, 2019.

[58] PetFinder and aschleg, "Cat breeds dataset. [Dataset]. Kaggle.com.." https://www.kaggle.com/datasets/ma7555/cat-breeds-dataset, 2019.

[59] C. Arteaga, "Interpretable machine learning with lime for image classification." https://nbviewer.org/url/arteagac.github.io/blog/lime_image.ipynb. Consultado: 15 de marzo de 2023.

[60] "Tensorflow 2.5 with GPU device (python 3.9, cuda 11.2.2 , cudnn 8.1.1) conda environment - windows 10." https://discuss.tensorflow.org/t/tensorflow-2-5-with-gpu-device-\python-3-9-cuda-11-2-2-cudnn-8-1-1-conda-environment-windows-10/1385, 2021. Consultado: 25 de marzo de 2023.

[61] "Procesos electorales 2023." https://consejouniversitario.buap.mx/sites/default/files/Procesos_Electorales_2023/HCU/Resultados_HCU_2023.pdf, 2023. Consultado: 10 de junio de 2023.

[62] S. Jamieson, "Escala likert." https://www.britannica.com/topic/Likert-Scale, 2022. Consultado: 05 de mayo de 2023.

[63] M. R. Saldaña, "Pruebas de bondad de ajuste a una distribución normal," *Revista Enfermería del Trabajo*, pp. 105–114, 2016.

[64] M. J. R. Hurtado and V. B. Silvente, "Cómo aplicar las pruebas paramétricas bivariadas t de student y anova en spss. caso práctico," *Revista d'Innovació i Recerca en Educació*, pp. 83–100, 2012.

A Primer apéndice: Códigos implementados

A.1 Código de implementación para el modelo de clasificación

Código 1 Definición del aumento de datos

```
|
|#Aplicamos la técnica de data augmentation.
|#Esto para evitar que el modelo procese la misma imagen más de una vez.
|
|data_augmentation = keras.Sequential(
|    [
|        #Volteo horizontal.
|        layers.RandomFlip('horizontal'),
|        #Aplicación de rotación con un factor de 0.1.
|        layers.RandomRotation(0.1),
|        #Aplicación de Zoom con un factor de 0.2
|        layers.RandomZoom(0.2)
|    ]
|)
```

Código 2 Implementación y definición de la base convolucional de la red VGG16

```
|#Implementando la base convolucional de la red VGG16, sin incluir o el clasificador.
|conv_base_vgg = keras.applications.vgg16.VGG16(
|    #Punto de control de pesos de la red VGG16
|    weights = 'imagenet',
|    include_top = False,)
|
|#"Congelamos" las capas del modelo preentrenado que no ajustaremos.
|#Mientras que las últimas 4 capas las dejamois para ajustarlas
|#con nuestro conjunto de datos.
|conv_base_vgg.trainable = True
|for layer in conv_base_vgg.layers[:-4]:
|    layer.trainable = False
```

Código 3 Implementación y definición del modelo utilizando transferencia de aprendizaje con VGG16

```
#Estructura del modelo "model_5animales".
#Parámetros de la imagen, 180 pixeles por 180 pixeles, con 3 canales.
inputs = keras.Input(shape=(180,180,3))
#Aplicamos aumento de datos a las imágenes de entrada.
x = data_augmentation(inputs)
#Preprocesa la imagen de entrada.
x = keras.applications.vgg16.preprocess_input(x)
# Se implementa la base convolucional de VGG sin el clasificador.
x = conv_base_vgg(x)
#Transformación de la red para adecuar la dimensión a un vector de 1xn.
x = layers.Flatten() (x)
# Creamos una capa densa de 256 de neuronas.
x = layers.Dense(256) (x)
# Ocupamos un dropout con una probabilidad de 0.5.
x = layers.Dropout(0.5) (x)
# Declaramos la salida de la red "5animales" con 5 neuronas (5 clases).
outputs = layers.Dense(5, activation='softmax') (x)
#-----------------------------------------------
#Definición del modelo (model_5animales)
model_5animales = keras.Model(inputs, outputs)
#-----------------------------------------------
```

Código 4 Compilación y entrenamiento del modelo

```
# Compilación del modelo, con la función de perdida, optimizador y métrica.
model_5animales.compile(loss = 'sparse_categorical_crossentropy',
              optimizer= keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics = ['accuracy'])
#Entrenamiento de la red neuronal.
#Especificando el conjunto de entrenamiento, épocas y conjunto de validación.
history = model_5animales.fit(
    train_dataset,
    epochs = 10,
    validation_data = val_dataset,
)

```

A.2 Código de implementación de LIME

Código 5 Implementación de algoritmo LIME

```
#Definición del modelo de clasificación que se utilizará.
model = tf.keras.models.load_model('model5animales.h5')
#Preprocesamiento de la imagen de entrada
url_img = 'name.jpg'
Xi = skimage.io.imread(url_img)
Xi = skimage.transform.resize(Xi, (180,180))
Xi = (Xi - 0.5)*2
#Predicción de la clase de la imagen de entrada
preds = model.predict(Xi[np.newaxis,:,:,:])
top_pred_classes = preds[0].argsort()[-2:][::-1]
#Extraer los superpíxeles de la imagen
superpixels = skimage.segmentation.quickshift(Xi, kernel_size=4,max_dist=200, ratio=0.2)
num_superpixels = np.unique(superpixels).shape[0]
#Creación de las pertubaciones aleatorias
num_perturb = 400
perturbations = np.random.binomial(1, 0.5, size=(num_perturb, num_superpixels))
def perturb_image(img,perturbation,segments):
    active_pixels = np.where(perturbation == 1)[0]
    mask = np.zeros(segments.shape)
    for active in active_pixels:
      mask[segments == active] = 1
    perturbed_image = copy.deepcopy(img)
    perturbed_image = perturbed_image*mask[:,:,np.newaxis]
    return perturbed_image
#Usar el clasificador para predecir las clases de las nuevas imágenes generadas.
predictions = []
for pert in perturbations:
  perturbed_img = perturb_image(Xi,pert,superpixels)
  pred = model.predict(perturbed_img[np.newaxis,:,:,:],verbose=False)
  predictions.append(pred)
predictions = np.array(predictions)
#Calcular distancias entre la imagen original y cada una de las imágenes perturbadas y calcular
#(importancia) de cada imagen perturbada.
original_image = np.ones(num_superpixels)[np.newaxis,:]
distances = sklearn.metrics.pairwise_distances(perturbations,original_image,
                                               metric='cosine').ravel()
kernel_width = 0.25
weights = np.sqrt(np.exp(-(distances**2)/kernel_width**2))
#Usar las perturbaciones, predicciones y pesos para entrenar un modelo de regresión lineal.
class_to_explain = top_pred_classes[0]
simpler_model = LinearRegression()
simpler_model.fit(X=perturbations, y=predictions[:,:,class_to_explain], sample_weight=weights)
coeff = simpler_model.coef_[0]
#Calcular características principales (superpíxeles).
num_top_features = 4
top_features = np.argsort(coeff)[-num_top_features:]
#Mostrar la explicación de LIME.
mask = np.zeros(num_superpixels)
mask[top_features]= True
img = perturb_image(Xi,mask,superpixels)
```

B Segundo apéndice: Encuestas realizadas para la evaluación de LIME

B.1 Encuesta para obtener los rasgos físicos que permiten identificar a un animal

Para observar la encuesta para obtener los rasgos físicos completa entrar al siguiente enlace: Encuesta Rasgos.

¿Qué rasgos físicos te permiten identificar más a un animal?

El cuestionario tiene la finalidad de realizar una nube de palabras de los rasgos físicos de los siguientes animales.

¿Para ti, que rasgos físicos de una araña permiten identificarla más (marca todos los que consideres pertinentes)? *

- [ ] Patas
- [ ] Cefalotórax
- [ ] Hileras
- [ ] Abdomen
- [ ] Quelíceros
- [ ] Otro: _____

B.2 Ejemplos de las encuesta para evaluar al algoritmo LIME

Para observar la encuesta tipo A para la evaluación de LIME entrar al siguiente enlace: Encuesta A.

Para observar la encuesta tipo B para la evaluación de LIME entrar al siguiente enlace: Encuesta B.

Encuesta para evaluar al algoritmo LIME

Te pedimos que nos ayudes a evaluar al algoritmo de Inteligencia Artificial explicable LIME.

Sección de la clase gato

Ten en cuenta que todas las imágenes de esta sección el modelo las clasificó como "Gato".
Si tienes dudas de los rasgos físicos consulta el siguiente link: Gato

Nota: Si tu respuesta es "otro:" responde el ¿Por qué?

Para la siguiente imagen dinos si la región o regiones que seleccionó LIME: *

- ◯ Corresponden a 1 o 2 palabras más frecuentes.
- ◯ Corresponden a 3 o 4 palabras más frecuentes.
- ◯ Corresponden a las 5 palabras más frecuentes.
- ◯ Al menos alguna de las que aparecen en la nube.
- ◉ No corresponden a nada de lo que aparece en la nube.
- ◯ Otro: _____

Sección de la clase perro

Ten en cuenta que todas las imágenes de esta sección el modelo las clasificó como "Perro".
Si tienes dudas de los rasgos físicos consulta el siguiente link: Perro

Nota: Si tu respuesta es "otro:" responde el ¿Por qué?

Para la siguiente imagen dinos si la región o regiones que seleccionó LIME: *

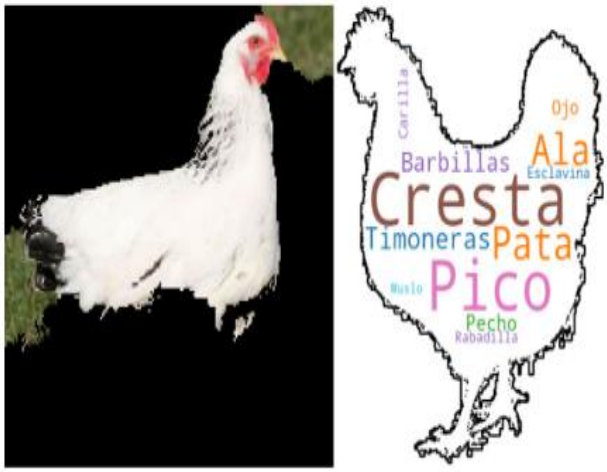
- ◯ Corresponden a 1 o 2 palabras más frecuentes.
- ◯ Corresponden a 3 o 4 palabras más frecuentes.
- ◯ Corresponden a las 5 palabras más frecuentes.
- ◯ Al menos alguna de las que aparecen en la nube.
- ◯ No corresponden a nada de lo que aparece en la nube.
- ◯ Otra...

Sección de la clase gallina

Ten en cuenta que todas las imágenes de esta sección el modelo las clasificó como "Gallina".
Si tienes dudas de los rasgos físicos consulta el siguiente link: Gallina

Nota: Si tu respuesta es "otro:" responde el ¿Por qué?

Para la siguiente imagen dinos si la región o regiones que seleccionó LIME: *



- Corresponden a 1 o 2 palabras más frecuentes.
- Corresponden a 3 o 4 palabras más frecuentes.
- Corresponden a las 5 palabras más frecuentes.
- Al menos alguna de las que aparecen en la nube.
- No corresponden a nada de lo que aparece en la nube.
- Otra...

Sección de la clase araña

Tener en cuenta que las imágenes de esta sección el modelo las clasificó como "Araña".
Si tienes dudas de los rasgos físicos consulta el siguiente link: Araña

Nota: Si tu respuesta es "otro:" responde el ¿Por qué?

Para la siguiente imagen dinos si la región o regiones que seleccionó LIME: *

- Corresponden a 1 o 2 palabras más frecuentes.
- Corresponden a 3 o 4 palabras más frecuentes.
- Corresponden a las 5 palabras más frecuentes.
- Al menos alguna de las que aparecen en la nube.
- No corresponden a nada de lo que aparece en la nube.
- Otra...

C Tercer apéndice: Estadísticos para prueba t

C.1 Resultados de homogeneidad de varianzas con *SPSS*

Prueba de homogeneidad de varianza

	Estadístico de Levene	gl1	gl2	Sig.
Se basa en la media	,405	1	12	,536
Se basa en la mediana	,338	1	12	,572
Se basa en la mediana y con gl ajustado	,338	1	11,732	,572
Se basa en la media recortada	,452	1	12	,514

Figura 38: Resultados de homogeneidad con la prueba *Levene*

C.2 Tabla de valores críticos de la distribución t

df	Nivel de significancia α				
	0.10	0.05	0.025	0.01	0.005
1	3.078	6.314	12.706	31.821	63.657
2	1.886	2.920	4.303	6.965	9.925
3	1.638	2.353	3.182	4.541	5.841
4	1.533	2.132	2.776	3.747	4.604
5	1.476	2.015	2.571	3.365	4.032
6	1.440	1.943	2.447	3.143	3.707
7	1.415	1.895	2.365	2.998	3.499
8	1.397	1.860	2.306	2.896	3.355
9	1.383	1.833	2.262	2.821	3.250
10	1.372	1.812	2.228	2.764	3.169
11	1.363	1.796	2.201	2.718	3.106
12	1.356	1.782	2.179	2.681	3.055
13	1.350	1.771	2.160	2.650	3.012
14	1.345	1.761	2.145	2.624	2.977
15	1.341	1.753	2.131	2.602	2.947
16	1.337	1.746	2.120	2.583	2.921
17	1.333	1.740	2.110	2.567	2.898
18	1.330	1.734	2.101	2.552	2.878
19	1.328	1.729	2.093	2.539	2.861
20	1.325	1.725	2.086	2.528	2.845
21	1.323	1.721	2.080	2.518	2.831
22	1.321	1.717	2.074	2.508	2.819
23	1.319	1.714	2.069	2.500	2.807
24	1.318	1.711	2.064	2.492	2.797
25	1.316	1.708	2.060	2.485	2.787
26	1.315	1.706	2.056	2.479	2.779
27	1.314	1.703	2.052	2.473	2.771
28	1.313	1.701	2.048	2.467	2.763
29	1.311	1.699	2.045	2.462	2.756
30	1.310	1.697	2.042	2.457	2.750
∞	1.282	1.645	1.960	2.326	2.576

Figura 39: Tabla de valores críticos de la distribución t [34]