ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

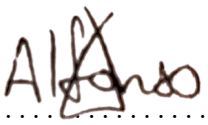
Evaluación de diferentes métodos XAI en modelos de visión por computadora
An evaluation of different XAI methods on computer vision models

Autor
Alfonso Rodrigo Cedillo Mayorga

Dirigido por
Sergio Núñez Covarrubias
Víctor Vaquero Soto

Madrid
May 2024

Alfonso Rodrigo Cedillo Mayorga, declara bajo su responsabilidad, que el Proyecto con título **Evaluación de diferentes métodos XAI en modelos de visión por computadora.**
An evaluation of different XAI methods on computer vision models. presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2023/24 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.:  Fecha: 21 / Junio / 2024

Autoriza la entrega:
Directores del Proyecto

Sergio Núñez Covarrubias

Víctor Vaquero Soto

Fdo.:  Fecha: 21 / junio / 2024

V. B. del Coordinador de Proyectos
Carlos Morrás Ruiz Falcó

Fdo.: ..... Fecha: ..... / ..... / .....

El autor D. Alfonso Rodrigo Cedillo Mayorga DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Evaluación de diferentes métodos XAI en modelos de visión por computadora, que esta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.
Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional: El trabajo contiene información confidencial, que no ha sido, ni será aprobada para su distribución por parte de la empresa donde se realizaron las prácticas y trabajo.

Índice

Dedicatorias y agradecimientos

Gracias a mis padres por el apoyo que me han dado durante el transcurso del trabajo, y a Víctor Vaquero por toda su ayuda con la elaboración del proyecto.

Resumen

La inteligencia artificial (en adelante, IA) y los algoritmos de aprendizaje automático están transformando radicalmente todos los sectores del mundo profesional y académico, volviéndose indispensables. Esta proliferación es impulsada no solo por avances tecnológicos, como los modelos de lenguaje de gran tamaño (LLMs), que han hecho el uso de la IA más accesible para usuarios no expertos, sino también por la necesidad de tomar decisiones informadas basadas en datos complejos. La democratización del acceso a la IA ha llevado a una adopción masiva, lo cual, a su vez, ha provocado una respuesta legislativa por parte de gobiernos/instituciones de todo el mundo [1] [2]. Estas nuevas leyes buscan salvaguardar los derechos humanos y asegurar que las decisiones tomadas por sistemas de IA en contextos de alto riesgo sean transparentes y comprensibles.

En este contexto emergente, la explicabilidad de la IA (XAI) gana importancia, convirtiéndose en un campo de estudio crucial para el desarrollo y la implementación ética de tecnologías de IA. Aunque la necesidad de explicabilidad es general en la IA, es particularmente crítica en el dominio de la visión por computadora, donde residen la mayoría de los modelos considerados como "cajas negras". Estos modelos, especialmente aquellos basados en técnicas de aprendizaje profundo, presentan una complejidad arquitectónica que desafía la interpretación directa de sus procesos y decisiones. Por lo tanto, el empleo de métodos de XAI en este ámbito no solo es deseable, sino esencial para desentrañar la lógica interna de los sistemas de IA, asegurando así su alineación con los principios éticos y las regulaciones emergentes. El entendimiento de la toma de decisiones de los modelos también nos permite ver biases anteriormente ignorados, permitiendo mejorar el rendimiento y toma de decisiones de los modelos, dejándonos identificar errores que no serían fáciles de identificar previamente.

Este estudio se enfoca en el campo de la visión por computadora, reconociendo que aquí se encuentran los desafíos más significativos y las oportunidades para avanzar en el desarrollo de técnicas de XAI. Al explorar y evaluar diferentes métodos de explicabilidad aplicados a modelos complejos de procesamiento de imágenes, aspiramos a contribuir a la creación de un marco de trabajo que no solo facilite la comprensión humana de las decisiones de IA, sino que también fomente una mayor confianza y adopción de estas tecnologías en aplicaciones críticas.

Glosario

IA/AI(inteligencia artificial): Se refiere a aquellos algoritmos que buscan imitar la inteligencia humana, permitiendo a las máquinas aprender de la experiencia y adaptarse a nueva información.

XAI(Explainable Artificial Intelligence): Se refiere a la rama dentro de la IA que busca desarrollar métodos y técnicas para que los modelos de IA sean comprensibles y transparentes, permitiendo entender y confiar los resultados generados por algoritmos de IA.

Explicabilidad: se refiere a la capacidad de un modelo para proporcionar razones detrás de sus decisiones de manera que los humanos puedan entender.

Interpretabilidad: implica la facilidad con la que se pueden comprender los procesos internos o resultados de un modelo.

Transparencia: aborda la claridad en la metodología, estructura y funcionamiento del modelo, permitiendo su evaluación y comprensión integral.

Explicación: En el contexto de algoritmos de IA, se refiere a responder por qué se ha conseguido un resultado (la respuesta puede variar según el objetivo). En el contexto de visión por computadora, se trata de resaltar aquellas partes de la imagen que más hayan influenciado en la decisión, y en un contexto de un modelo tabular, sería decir que features/inputs han tenido más efecto en la última decisión.

Explicación Local: Se trata de una explicación de la importancia del input para un caso en concreto. Por ejemplo, en el caso de imágenes, sería explicar una imagen en concreto.

Explicación Global: Se trata de una explicación de la importancia del input (en este caso de atributos/columnas) para un database completo. No existen las explicaciones globales para modelos de imágenes (solo una agregación de explicaciones locales).

Métodos Explicación Agnósticos: Se tratan de métodos de explicabilidad que pueden ser utilizados en todos los posibles modelos de IA, y que los tratan como si fuesen un "black box"(no se tiene acceso a su código).

Métodos Explicación Post-Hoc: Se trata de un estilo de explicabilidad que se refiere a métodos que se centran en analizar los datos solo después de haber sido modelados. Es decir, solo analizan los datos y el resultado que le aporta el modelo, no se considera ningún tipo de información interna del modelo.

Curva ROC: Es una representación gráfica de la sensibilidad frente a la razón de falsas alarmas en un clasificador binario.

AUC: Área debajo de la curva. Refiriéndose al área que se tiene en un grafo lineal.

AUROC: Área debajo de la curva ROC.

1. Introducción

La IA ha experimentado una evolución notable desde sus comienzos, marcando hitos significativos a lo largo de su historia. Desde la creación de la primera arquitectura de red neuronal en la década de 1940, la IA ha avanzado considerablemente, adoptando nuevas formas y técnicas. La introducción de los transformadores en 2017 supuso un punto de inflexión, revolucionando el procesamiento del lenguaje natural y estableciendo nuevas normas en el aprendizaje profundo.

Con el desarrollo de modelos de lenguaje de gran tamaño (LLMs), la IA ha alcanzado nuevas alturas en términos de capacidad y rendimiento, imitando cada vez mejor las capacidades humanas. Sin embargo, este progreso ha llevado a la creación de modelos cada vez más complejos y opacos, conocidos como "cajas negras", cuyo funcionamiento interno es difícil de interpretar.

2. Motivación

La nueva legislación sobre IA tanto en Europa como en Estados Unidos está impulsando medidas que van a implicar a los distintos sectores que hagan uso de IA a adoptar técnicas para explicar sus modelos [1] [2]. Estas técnicas no solo son útiles para cumplir con la legislación, sino que también deben ser vistas como un recurso valioso para mejorar la precisión de los modelos, al permitir una mejor comprensión de las razones por las que un modelo falla o acierta, o si toma decisiones correctas por razones equivocadas. Con la aparición de una multitud de técnicas de explicabilidad y la nueva legislación que la concierne, se empieza a considerar cómo comparar las diferentes técnicas para determinar cuál es la más adecuada para su uso con distintos modelos. Por ello, es necesario un estudio que evalúe con mayor certeza la adaptación de estas herramientas de explicabilidad.

Bajo la nueva legislación, la demanda de transparencia y reducción de sesgos en los sistemas de IA se intensifica significativamente. Esta legislación no solo impone estándares más estrictos sobre el funcionamiento de los sistemas de IA, sino que también establece lineamientos claros sobre cómo las empresas y los distintos sectores deben documentar y justificar las decisiones tomadas por estos sistemas. Se subraya la importancia de que cualquier modelo de IA, independientemente de su complejidad o área de aplicación, sea capaz de ofrecer explicaciones comprensibles y accesibles de sus resultados.

Ampliación del papel de las técnicas de explicabilidad

Las técnicas de explicabilidad, por lo tanto, adquieren una importancia renovada y más crítica bajo estos nuevos requisitos:

1. **Obligación de Explicaciones en Tiempo Real**: Los sistemas de IA deberán no solo actuar de manera justa y eficiente, sino también proporcionar explicaciones en tiempo real de sus decisiones.

2. **Desarrollo de Normas para la Explicabilidad**: Será necesario establecer normas claras sobre qué constituye una "explicación suficiente". Esto puede incluir detallar los factores que más contribuyeron a una decisión, el modelo de datos usado, y cualquier consideración ética tomada durante el diseño del sistema.

3. **Adopción de Herramientas Estándar de Explicabilidad**: Los proveedores de IA podrían tener que adoptar herramientas estandarizadas de explicabilidad que cumplan con los requisitos legales, asegurando así una base consistente para la evaluación y comparación de sistemas de IA.

4. **Capacitación y Conciencia**: La implementación efectiva de técnicas de explicabilidad requerirá también un esfuerzo significativo en capacitar a los profesionales de IA y a los usuarios de estos sistemas para entender y aplicar correctamente estas técnicas.

5. **Evaluación Continua y Feedback**: Será esencial no solo implementar estas técnicas al inicio, sino continuar evaluando y refinando las explicaciones basadas en el feedback de los usuarios y en los cambios en el entorno normativo y tecnológico.

Las técnicas de explicabilidad son fundamentales para cumplir con los nuevos requisitos de legislación de IA y para garantizar que la adopción de estas tecnologías avanzadas continúe de manera ética y responsable. A través de explicaciones claras y detalladas, los proveedores de IA no solo cumplirán con las regulaciones, sino que también fortalecerán la confianza del público en sus tecnologías.

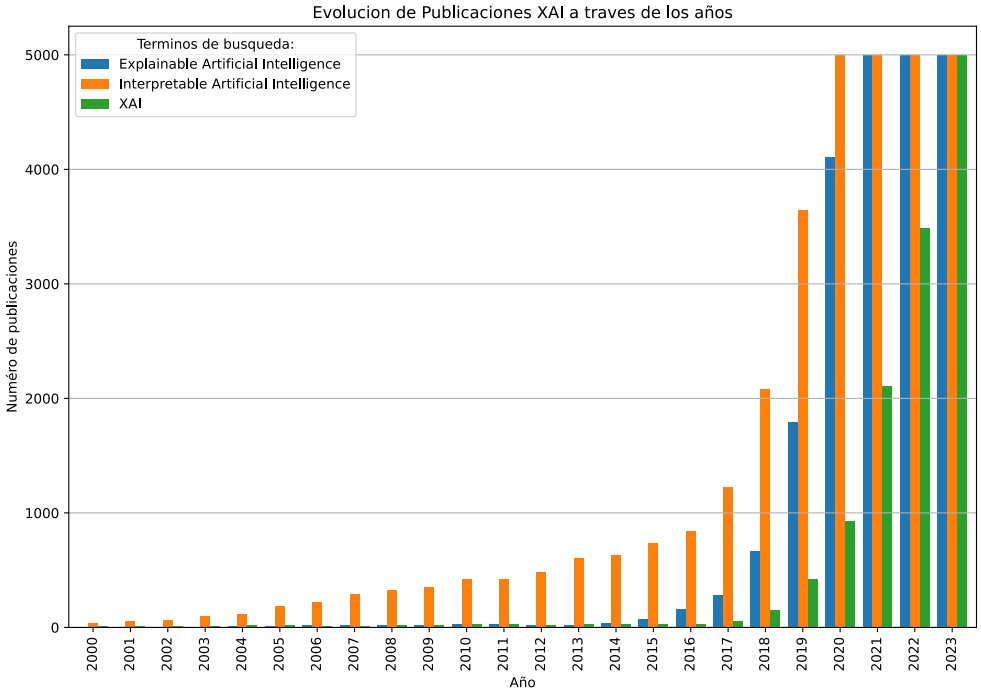


Figura 1: Popularidad de diferentes términos relacionados con explicabilidad en el contexto de IA extraída de Scopus por número de publicaciones desde el 2000 hasta 2023. Debido a un límite de resultados por cada query dentro del Scopus, se limita a 5k por término, pero dicho límite no impide que pueda apreciarse la tendencia en popularidad.

3. Background

La explicabilidad nace desde la idea básica de comprensión que necesitamos para cualquier sistema del que hacemos uso. En muchos de los modelos y técnicas de IA, no es necesario el uso de complejas técnicas de explicabilidad, ya que el modelo es explicable per se. Puede ser el ejemplo de una regresión lineal, donde los coeficientes de la regresión nos informan del peso que tiene cada una de las variables, o el caso de los árboles de decisión, que solo requieren mirar qué reglas de separación se han tomado para entender el razonamiento del modelo.
En la figura 2 del documento, podemos ver un árbol de decisión que clasifica datos basados en dos rasgos (o características). La lógica de este árbol de decisión se puede explicar de la siguiente manera: Primero, se evalúa si el Rasgo 1 es mayor que 3. Si no es así, la segunda decisión se toma en función del Rasgo 2, evaluando si es mayor o menor que 1. De aquí podemos explicar el proceso completo y comprender la decisión del modelo.

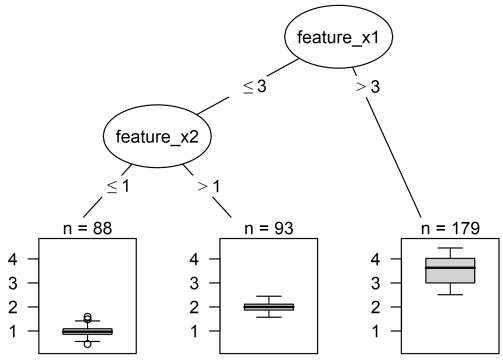


Figura 2: Árbol de decisión [3]

Desde que se empezó a popularizar el uso de modelos más complejos, se comenzaron a desarrollar muchas más herramientas para comprender a fondo su toma de decisiones. Desde aquí llegamos a las diferentes ramas de la explicabilidad y a una primera clasificación de modelos basada en dicha explicabilidad:

1. **Interpretables por diseño**: Existen modelos que son inherentemente interpretables, como los mencionados en el apartado anterior. Estos modelos no requieren ninguna herramienta especial.

2. **Cajas negras**: Un modelo de caja negra es un sistema cuyo funcionamiento interno es opaco o desconocido para el usuario.

Debido a la existencia de este tipo de modelos, se han desarrollado una multitud de diferentes métodos de explicabilidad. Desde algunos diseñados con un modelo especifico en mente, a aquellos que son agnósticos, y pueden ser aplicados a cualquier modelo de IA. Sin entrar en detalle de este tipo de técnica, solo recalcar, que los modelos agnósticos tienen la ventaja de ser compatibles con cualquier modelización de una librería, con tal de cumplir algunos requisitos(suficientes datos como para su estudio estadístico), mientras que los que no son agnósticos, suelen ser más rápidos y precisos en cuanto a explicaciones, con el requisito, de que la técnica tiene que poder dar acceso a información desde dentro del modelo (como puede ser la activación de una neurona en una red).

3.1. Técnicas de Explicabilidad

Las técnicas de explicabilidad o interpretabilidad pueden dividirse en bastantes ramas. Desde la manera en la que se consiguen los resultados, con algoritmos de análisis de perturbación o algoritmos intrínsecos que consiguen datos desde dentro del modelo, como podría ser el caso de un algoritmo que consiguiese las condiciones de separación de un árbol de decisión como puede ser el de la figura 2 o

los algoritmos de modelación subrogada, que se basan en entrenar un modelo interpretable para qué intenta llegar a los mismos resultados.

Otra forma alternativa de clasificar dichas técnicas, se basa en el tipo de explicación que proveen. Aquí podriamos distinguir explicaciones globales, de cohorte y locales. Para modelos de imágenes, solo son relevantes los modelos que tienen capacidad de explicaciones locales.

Técnicas de explicación global	Técnicas de explicación local
Importancia de características globales	LIME
SHAP	SHAP
Modelos sustitutos (surrogates)	Grad-CAM
	IG

Cuadro 1: Resumen de técnicas de explicabilidad global y local.

3.2. Explicación Global

La explicación global [4] [5] busca ofrecer una visión general de cómo el modelo toma decisiones. La mayor ventaja que presenta este tipo de técnicas es que permiten ver sesgos generalizados en un modelo, así como la importancia de cada característica individual a lo largo de todo el modelo.
Su mayor defecto es la gran duración empleada en generar dicha explicación, así como una generalización que no garantiza la ausencia de sesgos y ejemplos localizados.

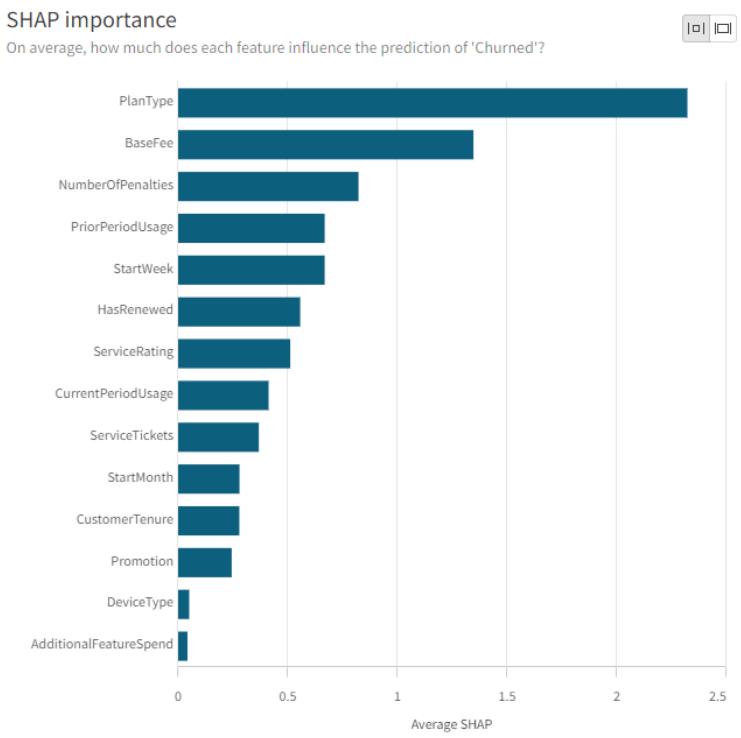


Figura 3: Ejemplo de explicabilidad global usando SHAP. [6]

3.3. Explicación de Cohortes

La explicabilidad de cohortes [4] [5]se refiere al proceso de entender e interpretar el comportamiento de un modelo de aprendizaje automático dentro de subconjuntos específicos de datos, conocidos como cohortes. En lugar de evaluar las explicaciones del modelo de manera individual o de todo el conjunto de datos, la explicabilidad de cohortes se enfoca en analizar patrones y similitudes en un grupo de instancias que comparten ciertas características o atributos.
La principal ventaja de este enfoque es que ayuda a identificar sesgos sistémicos, tendencias o conocimientos que pueden no ser evidentes al examinar predicciones individuales o incluso en análisis global. Mientras que la principal desventaja es el hecho de que no es posible automatizar este proceso, así como la posibilidad de seleccionar cohortes incorrectas, que conlleven conclusiones incorrectas sobre la presencia de sesgos. Si se analiza datos con sesgo, puede darse el caso de que no sea el modelo el que conduciendo a dichos sesgos, sino una selección incorrecta de datos que lleva a que el análisis estadístico otorgue una importancia indebida.

3.4. Explicación Local

Las técnicas de explicación local [4] [5], proporcionan explicaciones sobre predicciones individuales del modelo.
Se centran en explicar el espacio local de una sola predicción, es decir, solo explican una sola instancia de un dataset.
Su mayor ventaja es la velocidad de proceso, así como el hecho de tener explicaciones individualizadas para cada input, permitiendo estudiar de caso individual a caso individual, lo cual permite un análisis mucho más profundo de sesgos, no necesariamente generalizados.
Su mayor desventaja es el hecho de que dependen del algoritmo que se utilice, además, las explicaciones locales no son la mejor manera dé ver los sesgos de un modelo de manera general, al ser demasiado trabajo ir ejemplo a ejemplo. No todos los algoritmos permitirían que se pudiese realizar una explicación global a base de la agregación de las locales, con algunas explicaciones locales pudiendo equivocarse. [7] [8]

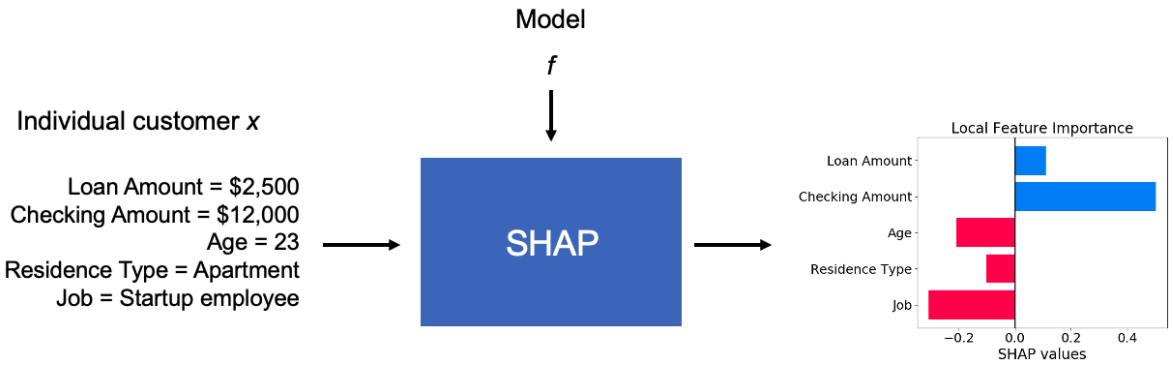


Figura 4: Ejemplo de explicabilidad local usando SHAP. [9]

A continuación, vamos a realizar una explicación a alto nivel de las principales técnicas usadas para llevar a cabo explicaciones globales y locales.

3.5. LIME

Local Interpretable Model-agnostic Explanations [8]. Se trata de uno de los métodos más populares para realizar explicaciones locales. Se trata de un modelo post-hoc, que realiza un análisis de

perturbación del input(solo de un dato). Para el caso de explicabililidad sobre algoritmos de visión por computadora, se consigue la importancia de los diferentes píxeles que forman una imagen.
LIME busca explicar las predicciones de un modelo complejo f alrededor de una instancia específica $\mathbf{x}$ utilizando un modelo simple g dentro de una vecindad definida. La explicación generada por LIME puede describirse como la solución al siguiente problema de optimización:

$$\xi(\mathbf{x}) = \underset{g \in G}{\operatorname{argmin}}\ L(f, g, \pi_{\mathbf{x}}) + \Omega(g)$$

donde:

- $\xi(\mathbf{x})$ es la explicación para la instancia $\mathbf{x}$ generada por LIME.
- f es el modelo complejo cuya predicción se desea explicar.
- g es el modelo simple que se utiliza para aproximar las predicciones de f localmente alrededor de $\mathbf{x}$.
- $L(f, g, \pi_{\mathbf{x}})$ es una función de pérdida que mide qué tan bien el modelo simple g aproxima el comportamiento de f en la vecindad de $\mathbf{x}$, ponderada por $\pi_{\mathbf{x}}$, una función de peso que enfatiza la importancia de las muestras cercanas a $\mathbf{x}$ para asegurar que la explicación sea local.
- $\Omega(g)$ es un término de complejidad que penaliza la complejidad del modelo simple g, promoviendo así la interpretabilidad de la explicación.
- G es el conjunto de modelos simples considerados para la aproximación.

LIME perturba la instancia $\mathbf{x}$ generando muestras en su vecindad, evalúa estas muestras con el modelo complejo f, y luego utiliza las evaluaciones junto con los pesos para entrenar el modelo simple g. Los coeficientes de g indican la importancia de cada característica en la predicción de $\mathbf{x}$, ofreciendo así una explicación local de por qué f realiza una predicción específica.

3.6. SHAP

SHapley Additive exPlanations [10]. Se trata de uno de los métodos más populares para realizar explicaciones locales y globales. Se trata de un modelo post-hoc, que realiza un análisis de perturbación del database entero, o por lo menos de una selección aportada por el usuario. Para el caso de explicabililidad sobre algoritmos de visión por computadora, se consigue la importancia de los píxeles. Lo único que difiere son los inputs para los diferentes algoritmos SHAP. Existen dos implementaciones distintas a la técnica de SHAP, Kernel SHAP y Deep SHAP, las cuales serán explicadas en el apéndice (Seccion 9.3),(Seccion 9.4)

SHAP se basa en el estudio de los SHAPLEY values/valores SHAPLEY, que es un método basado en la teoría de juegos:

$$\phi_i(f) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{i\}) - f(S) \right]$$

donde:

- $\phi_i(f)$ es el valor de SHAPley de la característica i para la función de predicción f.
- N es el conjunto de todas las características.
- S es un subconjunto de N que no incluye a i.
- $|S|$ es el número de características en el subconjunto S.

- $|N|$ es el número total de características.
- $f(S)$ es el valor de predicción del modelo para el subconjunto de características S.
- $f(S \cup \{i\})$ es el valor de predicción del modelo cuando la característica i se añade al subconjunto S.

3.7. IG

Integrated Gradients [11] es una técnica de interpretación local de modelos de aprendizaje automático que tiene como objetivo explicar las predicciones de los modelos basados en redes neuronales profundas.

El método IG aborda específicamente el desafío de atribuir la importancia de cada característica de entrada en la predicción de un modelo. Funciona generando un camino continuo en el espacio de entrada desde un punto de referencia (baseline) hasta el punto de entrada actual y calculando los gradientes de la salida del modelo con respecto a la entrada en varios puntos a lo largo de este camino. Luego, estos gradientes se integran (o acumulan) a lo largo del camino para obtener la importancia de cada característica.

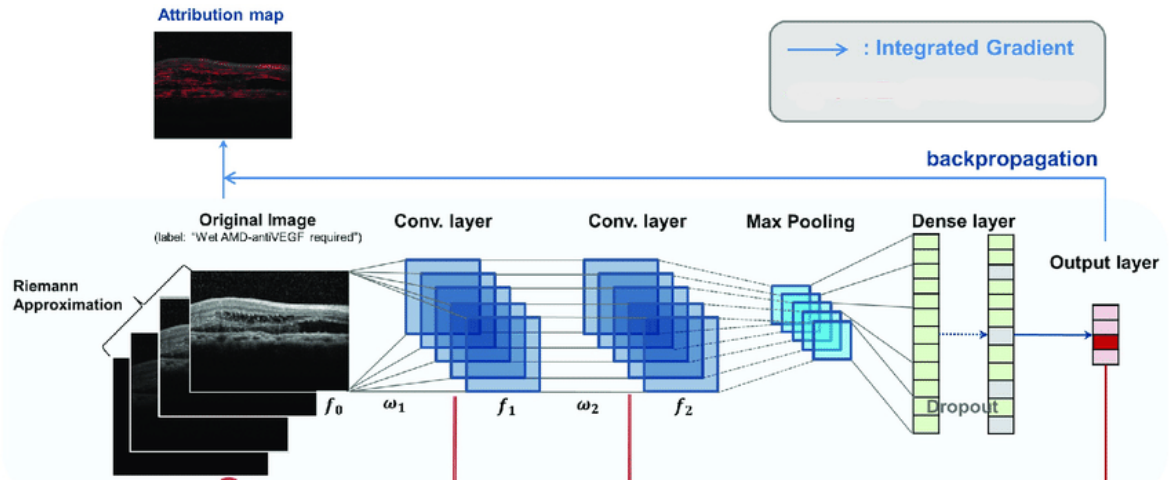


Figura 5: Proceso de Integrated gradients [12]

En el caso concreto de modelos de visión por computadora, se consigue la importancia de cada pixel en la imagen a lo largo de las diferentes capas de la red, y se agregan en una importancia final. La fórmula para calcular la atribución de la característica i es:

$$\mathrm{IG}_i(f, x) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

donde:

- $\mathrm{IG}_i(f, x)$ es la atribución de Integrated Gradients para la característica i en la instancia de entrada x con respecto a la función de predicción f.
- x es la instancia de entrada para la cual queremos explicar la predicción del modelo.
- x' es el punto de referencia (baseline) sobre el cual se compara la instancia de entrada x. Típicamente, x' es una entrada que se espera que tenga una predicción baja o neutra.

- x_i es el valor de la característica i en la instancia de entrada x.
- x'_i es el valor de la característica i en el punto de referencia x'.
- $\frac{\partial f(x)}{\partial x_i}$ es el gradiente de la función de predicción f con respecto a la característica i, evaluado en la entrada x.
- α es un escalar que varía de 0 a 1, utilizado para interpolar entre el punto de referencia x' y la instancia de entrada x.

Este método cumple con la propiedad de sensibilidad y especificidad, asignando importancia cero a las entradas que no cambian la predicción y distribuyendo toda la diferencia en la predicción entre las entradas basadas en sus gradientes.

Figura 6: Ejemplo de Intregrated gradients sobre uno de los datasets empleados en el presente estudio. A la izquierda, puede observarse la imagen original, mientras que a la derecha se trata de la importancia otorgada por IG, donde se puede apreciar como resalta en especial los bordes del objeto presente en la imagen, en este caso, indicando que es lo que más le importa a la capa para el ejemplo.

3.8. Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) [13] es un método de visualización que ayuda a entender qué partes de una entrada fueron consideradas importantes por una red neuronal convolucional (CNN) para la decisión de una predicción específica. El proceso empleado por dicha técnica consiste en:

- **Selección de la capa de interés:** Primero, se elige una capa de la CNN que capture las características relevantes para la tarea de predicción. Generalmente, esta es una de las últimas capas convolucionales.
- **Cálculo de gradientes:** Luego, se calculan los gradientes de la clase objetivo (es decir, la clase de salida para la cual se quiere explicar la predicción) con respecto a los mapas de características de la capa seleccionada. Estos gradientes indican la importancia de cada mapa de características para la clase objetivo.
- **Ponderación de los mapas de características:** Los gradientes se promedian a lo largo de las dimensiones de ancho y alto de la capa, dando como resultado un peso para cada mapa de características.
- **Combinación de mapas de características:** Los mapas de características se ponderan por los pesos calculados y se combinan para obtener un heatmap(mapa de calor) de la misma dimensión espacial que la capa de interés. Este heatmap resalta las regiones importantes para la predicción de la clase.

Desde el punto de vista matemático, el proceso anterior se puede formular de la siguiente manera:

$$\text{Grad-CAM}_i(f, x) = \text{ReLU}\left(\sum_k \frac{\partial Y^c}{\partial A_{ij}^k}\right)$$

donde:

- $\text{Grad-CAM}_i(f, x)$ es la atribución de Grad-CAM para la característica i para la instancia de entrada x con respecto a la función de predicción f.
- Y^c es la salida de la clase c antes de la capa de softmax.
- A_{ij}^k es la activación de la característica espacial en la capa convolucional k en la posición (i, j).

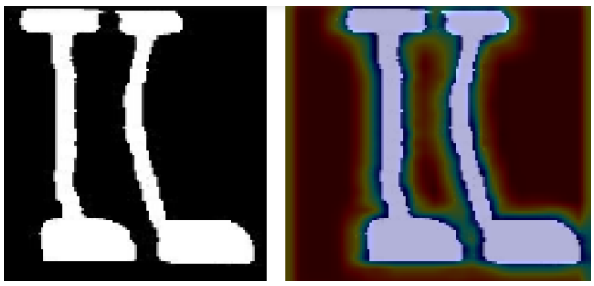


Figura 7: Ejemplo de Grad Cam en uno de los datasets. A la izquierda mostramos la imagen original, mientras que en la derecha se encuentra la explicabilidad de cada pixel otorgada por Grad Cam, observándose como se ha centrado en la figura central, así como los bordes de esta misma en este ejemplo.

4. Metodología

4.1. Métricas

Con vistas a un mayor uso de las diferentes técnicas de explicabilidad en IA impulsadas por las diferentes legislaciones, es imprescindible comenzar a normalizar el uso de métricas comunes para poder tener un estándar conjunto. Uno de los principales problemas presentes en el campo XAI, es que existe mucho contenido no unificado, por lo que existen algoritmos que aun siendo interesantes, no es posible saber cuál es su verdadero rendimiento en comparación a los más comunes.
La importancia de utilizar métricas estandarizadas para medir las técnicas en un campo emergente como el XAI radica en varios factores clave. Primero, las métricas comunes permiten una evaluación objetiva y consistente del rendimiento de los algoritmos. Sin estándares claros, los investigadores y desarrolladores pueden interpretar los resultados de manera diferente, lo que dificulta la comparación directa y la replicabilidad de los estudios.
Por este motivo, se ha llevado a cabo un análisis de distintas fuentes [14–23], donde presentan métricas aplicables en el contexto de XAI. A continuación, presentamos las métricas más relevantes que permiten llevar a cabo la estandarización comentada anteriormente e incluimos ciertas métricas que hemos descartado y la justificación de dicho descarte.

4.1.1. Recursos Utilizados

Medir el tiempo de ejecución, así como cuanta memoria usa cada método. Esto nos permitirá medir como de eficientes son las diferentes técnicas al conseguir las explicaciones. Este tipo de métricas se

deberán tener en cuenta para modelos complejos con muchos datos, al igual que se empieza a tener en cuenta para modelos de IA generativa. A gran escala, el uso de un método un poco menos preciso, pero mucho más eficiente, puede ayudar económicamente desde el punto de vista de uso de recursos, así como medioambientalmente, ayudando a la empresa a reducir su huella de carbono al reducir su uso de electricidad.
Aparte de tiempo, esta función también mide el número de funciones que se utilizan, así como funciones primitivas, con el objetivo de saber cuan eficientemente está escrito el código, y saber si hay ineficiencias muy grandes, por lo menos desde la perspectiva de sobreúso de llamadas a funciones.

Para esto se utilizarán las librerías en Python de resource, cProfile [24] y memory_profiler [25] . Con ellas se puede tener un registro del uso de CPU con cuanto tiempo es utilizado, el uso de memoria, así como el número de llamadas dentro del código.

Estos tests se realizaron 10 veces por método, con un reinicio de kernel realizado cada vez que se ejecutó. La memoria utilizada es la memoria antes y después de utilizar el método.

Algorithm 1 Resource and Performance Profiling Algorithm

function TIME RESOURCE MEASUREMENT FUNCTION
    $Profiler \leftarrow Initiate$
    $start\_cpu \leftarrow cpu\_profile$
    $x \leftarrow explainer.\_instance(image\_to\_explain, predict, top\_labels)$
    $end\_cpu \leftarrow cpu\_profile$
    $cpu\_time\_used \leftarrow end\_cpu - start\_cpu$
    $Numberofcalls \leftarrow Profiler$
end function
function MEMORY RESOURCE MEASUREMENT FUNCTION
    $MemoryProfiler \leftarrow Initiate$
    $mem\_usage \leftarrow memory\_usage.\_instance(explainer$
        $interval = 0{,}01, max\_usage)$
end function

4.1.2. Estabilidad

Mide la variación de las explicaciones ante pequeñas perturbaciones dentro de los datos. Para algoritmos de visión por computadora, dichas perturbaciones son aplicadas sobre una misma imagen. En algunos estudios, también se le conoce con el nombre de robustez.

La metodología se basará en añadir ruido gaussiano a las imágenes, donde se comprobará y medirá que las explicaciones no varían de manera significativa en comparación con la explicación sobre la imagen original. La metodología empleada está basada en el estudio presentado en la referencia [14]. Para asegurar la normalización en todos los modelos, así como métodos, primero se realizará un proceso donde se asignará el valor más alto de la explicación a 100, y el menor a 0. Esto garantizará valores homogéneos para todos los métodos. Se hará mediante la siguiente normalización:

$$\text{new\_value} = \left(\frac{x - \min \times (b-a)}{\max - \min}\right) + a$$

En cuanto a las métricas específicas de estabilidad, se han empleado:

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\|\|B\|}$$

El Cosine similarity mide el ángulo coseno entre dos vectores en un espacio multidimensional. Es una medida de orientación, no de magnitud, con rango de -1 a 1, con 1 significando que tienen la

misma orientación, y -1 con vectores en direcciones exactamente opuestas, y 0 significando ortogonalidad con independencia de la Magnitud. Esto es especialmente útil, porque en todos los modelos, los resultados son matriciales, y pueden variar dependiendo de los cambios.

También se ha empleado como métrica el coeficiente de correlación de Pearson:

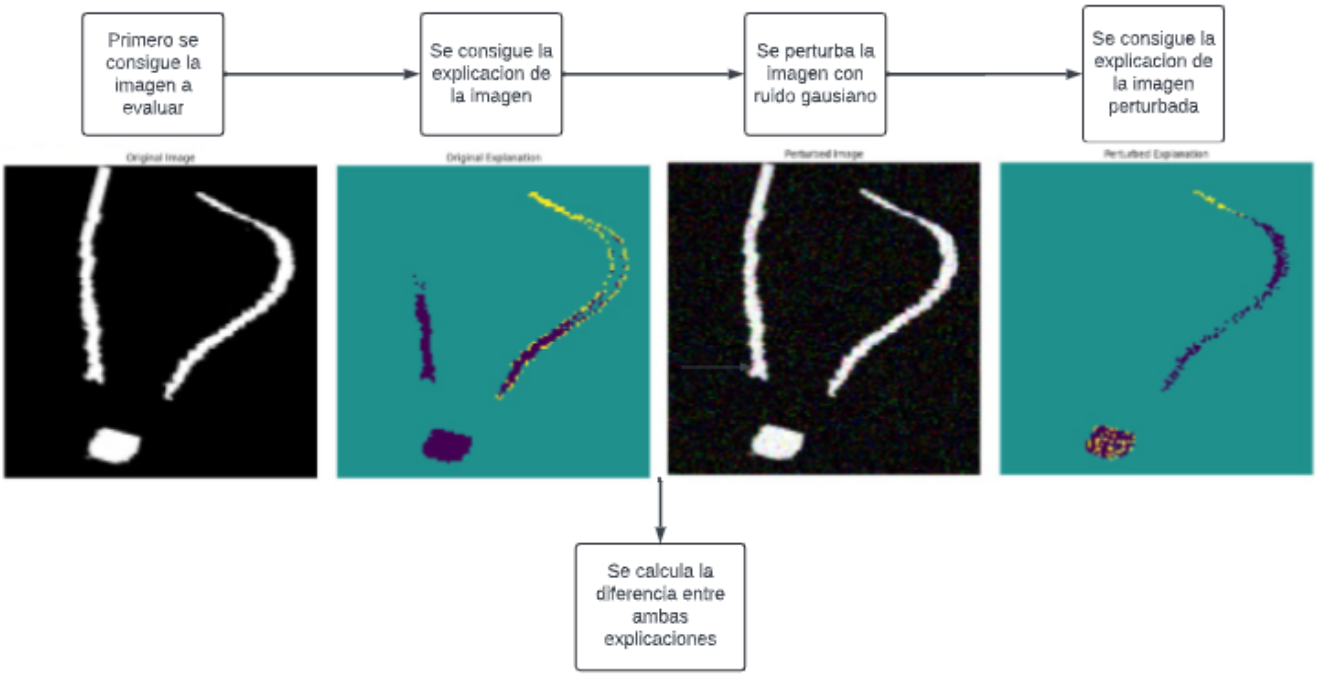
$$\text{Pearson Coefficient} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

El coeficiente de correlación de Pearson mide la correlación lineal entre dos variables devolviendo un valor entre -1 y 1. Es útil para detectar relaciones lineales donde un cambio en una variable está asociado con un cambio proporcional en otra variable. En este caso, nos sirve para ver la relación entre las matrices de resultados que recibiremos, pudiendo detectar que sigan estando relacionadas. Un valor de 1 conlleva una relación positiva lineal perfecta, y -1 una relación lineal negativa perfecta, con 0 significando la ausencia total de relación lineal.
En cuanto a otras métricas de similitud comunes, como distancia euclidiana y de Manhattan, no se han empleado porque son muy propensas a cambiar demasiado debido a magnitud, sin ser el caso. En cuanto al índice de Jacquard no es factible su uso debido al tipo de datos, ya que Jacquard se integra mejor con datos binarios que continuos, como es el caso de este estudio.

Como hemos mencionado anteriormente, la estabilidad también se llama robustez/robustness en algunos escritos, aunque en este estudio denominaremos robustez a otra métrica mencionada posteriormente. Existen también implementaciones muy similares como la propuesta en Towards the Unification and Robustness of Perturbation and Gradient Based Explanations: [18], donde se utiliza Lipschitz continuity para medir la estabilidad/robustez.

Algorithm 2 Image Explanation and Perturbation Analysis

```
function STABILITY(image)
    original_explanation ← func(image)
    for i do
        perturbed_image ← add_gaussian_noise(image)
        perturbed_explanation ← explainer(perturbed_image)
        x ← compute cosine similarity
        y ← compute Pearson correlation
    end for
end function
```



4.1.3. Fidelidad

Se basa en medir cómo de bien se acerca la explicación de los algoritmos a los modelos. Para el caso de algoritmos de visión por computadora, esta medición se lleva a cabo insertando o quitando los píxeles más importantes para la clasificación de esa imagen, y viendo si cambia la clasificación o el resultado de la misma. En caso de que sea una buena explicación, entonces tendrá la máxima área debajo de la curva en inserción, y el mínimo en la curva de deleción. Cuando se insertan píxeles, cuantó antes pueda llegar al resultado de la explicación de la imagen completa, mejor se considera la explicación.
Se harán los test, utilizando las métricas de inserción y deleción, donde se conseguirá un porcentaje de los píxeles más importantes de las imágenes y desde ahí se añadirán o quitaran y se comprobará cuanto porcentaje de los píxeles con mayor importancia en la explicación son necesarios para cambiar el resultado.
En el presente estudio, para la deleción e inserción, los píxeles faltantes (los cuales todavía no se han

insertado o ya se han eliminado) se representarán mediante píxeles negros, por lo que una imagen vacía será aquella que es negra completamente. Esto podría cambiar dependiendo del dataset sobre el que se trabaje, pero es la aproximación utilizada en este caso, ya que ningún modelo usado acepta píxeles nulos.

$$\text{Fidelity} = \frac{N}{100}$$

siendo N el número de píxeles necesarios, y 100 representando la imagen entera.
La cercanía a la solución final se representará mediante una función de cercanía, donde se medirá la distancia máxima desde el objetivo (resultado final) y después se medirán las distancias basadas en los porcentajes, con 0 por ciento siendo la distancia más lejana y 100 siendo el resultado final. Esto se hace para normalizar los datos.

Algorithm 3 Algoritmo de Inserción

```
function InsertExplainability(image, model)
    resultados ← []
    for porcentaje ← 0 to 100 step 2,5 do
        num_pixeles ← ⌊(porcentaje / 100) × total_pixeles(image)⌋
        insertion_mask ← create_insertion_mask(image, num_pixeles)
        perturbed_image ← apply_mask(image, insertion_mask)
        resultado ← model.predict(perturbed_image)
        append(resultados, resultado)
    end for
    return resultados
end function
```

Algorithm 4 Algoritmo de Deleción

```
function DeleteExplainability(image, model)
    resultados ← []
    for porcentaje ← 0 to 100 step 2,5 do
        num_pixeles ← ⌊(porcentaje / 100) × total_pixeles(image)⌋
        deletion_mask ← create_deletion_mask(image, num_pixeles)
        perturbed_image ← apply_mask(image, deletion_mask)
        resultado ← model.predict(perturbed_image)
        append(resultados, resultado)
    end for
    return resultados
end function
```

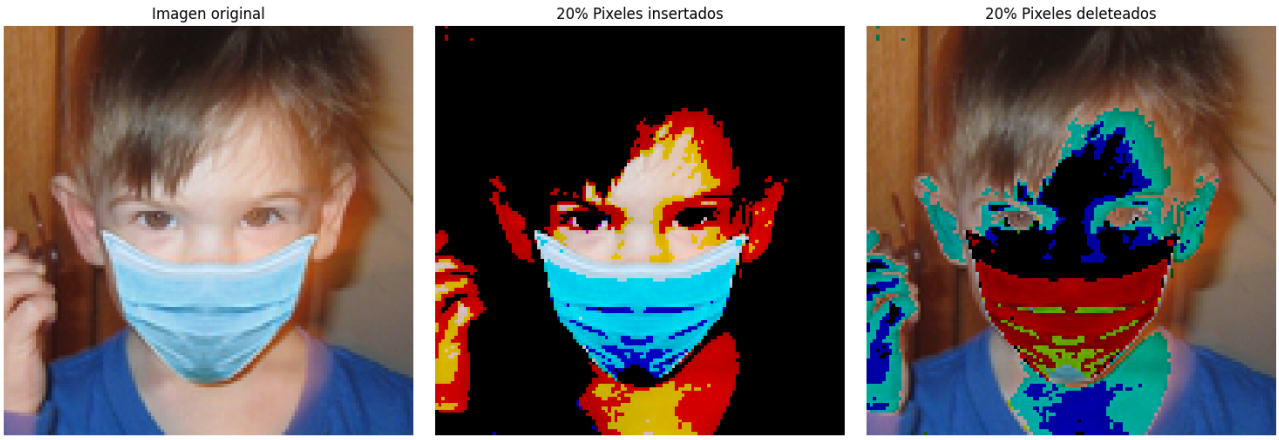


Figura 9: Demostración de inserción y deleción con 20 por ciento y aplicando la técnica de LIME sobre uno de los datasets de estudios empleados en el presente trabajo.

4.1.4. Métricas no utilizadas

Robustez: La robustez ante ataques adversariales sigue siendo un campo de estudio amplio dentro de la explicabilidad, y los métodos que blindan a los algoritmos de explicación de estos ataques, suelen bajar mucho las calidades de las explicaciones. Es por ello que hasta que no se consigan mejoras significativas, no consideramos que sea un punto importante a evaluar la robustez de algoritmos. En un futuro, medir su rendimiento será necesario, pero por ahora se deben desarrollar técnicas mejores y más reconocidas. [21] [23]

Métricas de fidelidad basadas en explicaciones humanas: Intersection over Union (IoU), es una métrica que se basa en medir la intersección entre 2 sets, en este caso sería entre dos imágenes, una siendo el mapa de importancia sacado de la técnica de explicabilidad, con una imagen anotada por humanos. Se ve su uso en XAI Benchmark for Visual Explanation [22]
Es una métrica interesante, pero no consideramos que sea una métrica válida en el contexto de algoritmos explicables, ya que estos no deben ceñirse a aquello anotado por un humano, sino a aquella parte de los datos que son significados en las decisiones del modelo. No debería recompensarse resaltar partes de la imagen que puedan interesar a un especialista, si el modelo realmente se está fijando en otros atributos. Esta métrica solo debería usarse en el caso de evaluar modelos de visión, no algoritmos de explicabilidad.
Las otras métricas mencionadas en las distintas fuentes revisadas son: pointing game, que calcula qué parte de la explicación recae sobre el área resaltada por humanos y después Precision, Recall y F1 Score. Todas usadas para la comparación entre explicaciones humanas y explicacions de técnicas XAI. Debido a la subjetividad aportada por estas métricas, se decidió que no deberían ser métricas consideradas para crear el estándar de este escrito.

4.2. Bases de datos utilizadas

En primer lugar, trabajamos con una base de datos y modelo interno de Repsol, englobados ambos en un caso digital conocido con el nombre de OCCULARIS. Se trata de un modelo que busca encontrar los desperfectos que existen dentro de los soportes de seguridad de los camiones de butano y cuya conceptualización se ha llevado a cabo en la factoría de GLP situada en Pinto.

4.2.1. Flujo del modelo OCCULARIS

Este modelo consta de 3 partes:

- Una primera parte de detección de objetos, dondé se busca detectar donde se encuentran los "latiguillos"de seguridad que aseguran el producto en el camión. Dichos objetos se detectan sobre un video que registra el paso de los camiones en las factorías. Además de detectar los latiguillos, también el número de ellos que hay, con cada uno de ellos teniendo un número asignado para identificarlos. Dicho proceso puede observarse en la imagen 11.
- Una segunda parte de máscaras que segmenta esos objetos. Dicho proceso puede observarse en la imagen 12.
- Una tercera parte de detección de anomalías que busca detectar latiguillos en mal estado. Dicho proceso puede observarse en la imagen 13 y 14.

El objetivo del sistema en producción es que pasen los distintos camiones que transportan el producto y mediante una cámara que registra su salida de la factoría, se registren imágenes de los objetos de interés, y a partir de ahí, pasarlos a los modelos y llevar a cabo una clasificación que determine si se encuentra en buen estado el latiguillo o si necesita ser revisado antes de proceder al transporte del producto.

Figura 10: Ejemplo de imagen de video recibida y con la detección realizada. Los distintos latiguillos detectados están resaltados en rojo.

Figura 11: El objeto que se quiere conseguir

Desde aquí, se utiliza un modelo para realizar una segmentación del modelo, sacando la silueta del latiguillo.

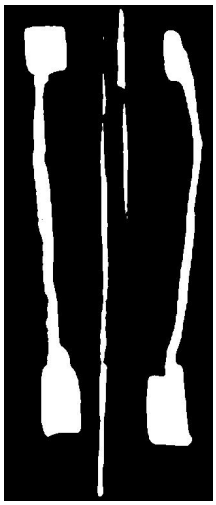


Figura 12: El objeto segmentado, para posterior detección de anomalías (ejemplo clasificado como anómalo).

En el presente trabajo nos hemos centrado en el algoritmo de detección de anomalías sobre el cual aplicaremos diferentes técnicas de explicabilidad y analizar su comportamiento con las métricas presentadas anteriormente. Hemos seleccionado esta parte del flujo, ya que es la más crítica de todo el proceso y donde comprender cuáles son las partes más importantes de la imagen donde el modelo se está fijando en la toma de decisiones y donde además se necesita llevar a cabo una justificación del motivo por el que el latiguillo se consideró que está en condiciones anómalas.
Se realizó una ampliación del dataset después de varios meses, con un nuevo tipo de segmentación realizado. Se tienen alrededor de 600 imágenes, con las siguientes características:

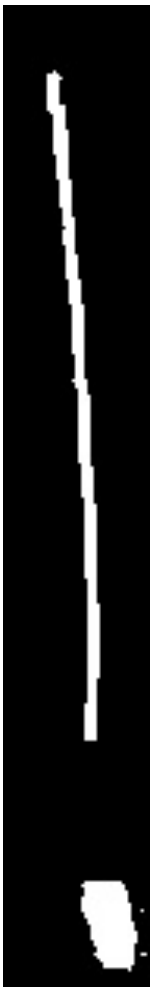


Figura 13: Imagen considerada sin anomalía.

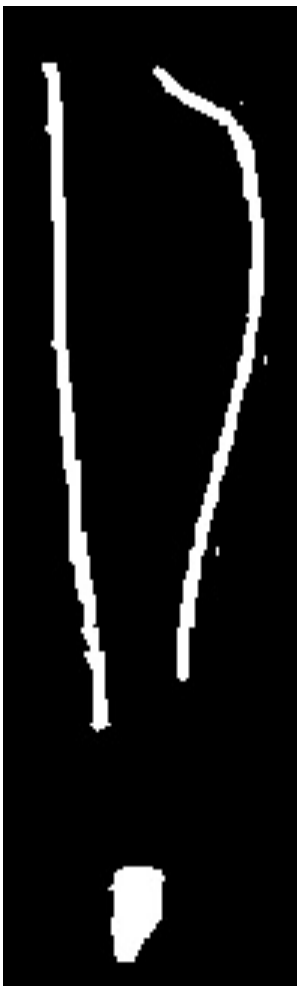


Figura 14: Imagen anómala 1.

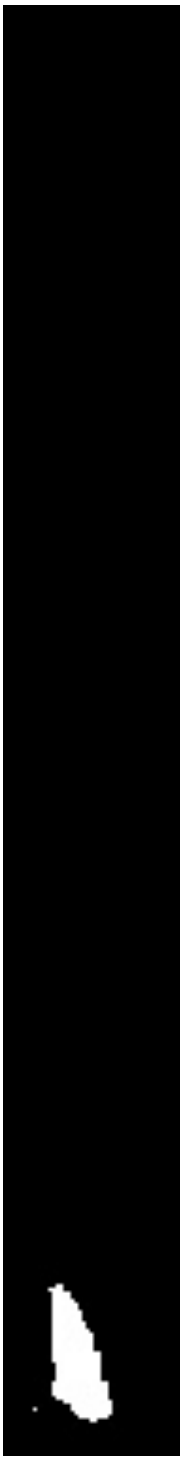


Figura 15: Imagen anómala 2

Figura 16: Imagen anómala 3

Las imágenes normales se tratan de latiguillos individuales, o parejas de estos, como se muestra en la figura 13. En cambio, las anomalías pueden ser muy diferentes, desde la falta del punto inferior, a la ausencia del latiguillo por completo, como se pueden ver en los ejemplos anteriores.

4.2.2. Autoencoder

Para llevar a cabo el proceso de detección de anomalías sobre los latiguillos, dentro de la compañía ya se disponía de un modelo para la detección de anomalías sobre imágenes, basado en la librería de anomalib. Dicho modelo tenía la particularidad de tener una estructura muy específica y no compatible con casi ninguna librería de explicabilidad (únicamente LIME se podía ejecutar correctamente sobre el modelo).

Es por ello que se investigó y se desarrolló un modelo alternativo, con más documentación, y resultados comparables. Se basó en la arquitectura de un autoencoder en torch, desde el cual se implementó una función compuesta basada en Binary Cross Entropy(Entropía Cruzada Binaria), donde se consigue la función de Binary cross entropy de las imágenes no anómalas, y se premia que tenga un loss bajo, mientras que se introducen también las imágenes anómalas, y se penaliza su construcción correcta. De esta manera se llegó a obtener un modelo con AUROC de 0.96 y un ImageF1 score de 0.92. Dichos resultados mejoraban el modelo implementado con anomalib.

Con el desarrollo de este modelo, se consiguió crear un modelo más interpretable en comparación con el de la compañía, siendo a su vez mucho más eficiente en el uso de recursos que el original. Su principal atractivo siendo que este es capaz de ejecutar operaciones en batch, al contrario que el modelo de anomalib, que hacía que fuese mucho más lento.

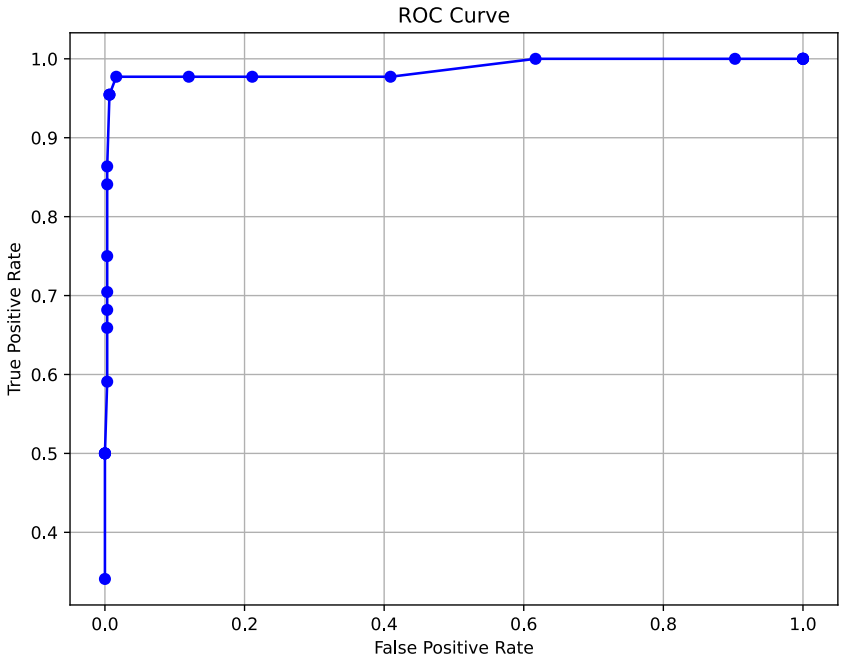


Figura 17: ROC del modelo autoencoder desarrollado en el presente trabajo para la detección de anomalías.

En la siguiente tabla se presenta el proceso de entrenamiento del modelo de autoencoder y el flujo desarrollado para llevar a cabo la detección de anomalías.

Algorithm 5 Autoencoder Training with BCE Margin Loss and Anomaly Detection with BCE

function TrainAutoencoder(normalImages, anomalousImages, epochs, margin)
    for each epoch **do**
        Initialize normalDataLoader with normalImages
        Initialize anomalousDataLoader with anomalousImages
        for each batch from normalDataLoader and anomalousDataLoader **do**
            $normalOutputs \leftarrow$ Compute the reconstruction of $normalBatch$
            $anomalousOutputs \leftarrow$ Compute the reconstruction of $anomalousBatch$
            $loss \leftarrow bce_margin_loss()$Compute the loss with both reconstructions
            Backpropagate the loss and update model weights.
        end for
    end for
end function
function DetectAnomalies(autoencoder, images, threshold)
    for each image in images **do**
        $reconstruction \leftarrow$ Compute the reconstruction of the image
        $loss \leftarrow$ Calculate BCE loss between the image and its reconstruction
        if loss $>$ threshold **then**
            Mark image as an anomaly
        end if
    end for
end function
function BCEMarginLoss(normalOutputs, normalImages, anomalousOutputs, anomalousImages, margin)
    $normalLoss \leftarrow$ F.binary_cross_entropy(normalOutputs, normalImages)
    $anomalousLoss \leftarrow$ F.binary_cross_entropy(anomalousOutputs, anomalousImages)
    $loss \leftarrow normalLoss + F.relu(margin - (anomalousLoss - normalLoss))$ **return** $loss$
end function

4.2.3. Flujo del modelo base de datos adicional de mascarillas

Para realizar una comparación adecuada entre los algoritmos de explicabilidad, se decidió utilizar una base de datos adicional. Esto es necesario para obtener información libre de sesgos específicos del modelo de autoencoder. La utilización de datos y modelos adicionales ofrece varias ventajas:

- **Diversidad de Datos**: Garantiza una evaluación más robusta y generalizable, evitando que los resultados estén limitados a las características de un solo conjunto de datos.
- **Reducción de Sesgos**: Minimiza los sesgos específicos de un único modelo o conjunto de datos, proporcionando una visión más equilibrada del rendimiento de los algoritmos.
- **Validación Cruzada**: Permite contrastar resultados en diferentes contextos, fortaleciendo la validez de las conclusiones.
- **Robustez del Modelo**: Facilita la evaluación de los algoritmos en diversas arquitecturas y configuraciones, mejorando la comprensión de su comportamiento.

En conclusión, la inclusión de una base de datos y un modelo adicionales es esencial para una evaluación exhaustiva y precisa de los algoritmos de explicabilidad, asegurando resultados robustos y generalizables.

El segundo modelo se trata de un modelo de clasificación multi-clase, centrándose en la correcta o incorrecta posición o completa ausencia de mascarillas sobre el rostro de personas. Es un modelo

público, con un objetivo claro, y un dataset extenso, por lo que se eligió para este estudio. Es un modelo bastante menos complejo que el modelo de anomalías, y en parte por lo que se escogió, ya que en este modelo, un cambio menor no afecta demasiado al resultado, como puede ser el caso del modelo de anomalías.
Se trata de un CNN preentrenado y disponible en la siguiente fuente [26]. El dataset es parte de los datasets públicos de mldatasets, con el nombre:"maskedface-net_thumbs_sampled".

Figura 18: Ejemplo de persona con máscara.

Figura 19: Ejemplo de persona con incorrecta posición de la máscara.

Figura 20: Ejemplo de persona sin máscara.

5. Resultados

En primer lugar, mostramos los datos de la máquina utilizada en el presente estudio. En caso de que se quieran recrear los experimentos.

VM Size	Standard D8ds v5	vCPU	8 cores
RAM	32 GiB	Operating System	Linux
Image Publisher	Microsoft	Image Offer	Ubuntu 20.04
Image Plan	2004-gen2	VM Generation	V2
VM Architecture	x64	CPU Model	Intel(R) Xeon(R) Platinum 8370C CPU @ 2.80GHz
Cores per socket	4	Threads per core	2
L1d Cache	192 KiB	L1i Cache	128 KiB
L2 Cache	5 MiB	L3 Cache	48 MiB
CPU Speed	2.80 GHz	Virtualization	VT-x
Online Cores	0-7		

Cuadro 2: Características de la máquina empleada donde se ejecutaron los diferentes experimentos.

Por cada métrica empleada, se llevaron a cabo distintos experimentos.

1. **Recursos:**
   a) Se realizaron dos experimentos diferentes, cada uno con 10 iteraciones para reflejar la media estadística.
   b) Se midieron el tiempo de CPU, Function Calls y Primitive Calls. Cada algoritmo se ejecutó 10 veces, obteniendo la media para cada métrica.
   c) Se realizó el mismo procedimiento para medir el uso de memoria.
   d) Siempre se utilizó la misma imagen para cada ejecución, garantizando que no existiera ningún tipo de sesgo externo.
2. **Estabilidad:**
   a) Para asegurar consistencia, se eligieron las mismas imágenes para todos los tests y se generó el ruido gaussiano con los mismos parámetros exactos.
   b) Se realizaron pruebas en 6 imágenes diferentes en Máscaras, con 2 de cada categoría, y 8 para OCCULARIS, 4 de cada categoría.
   c) Se realizaron 2 perturbaciones por imagen, y se ejecutó en un loop de 5 iteraciones.
   d) Esto garantizó que no hubiera una estabilidad indebida en los explicadores con aleatoriedad, y aseguró que incluso una ejecución separada, que haya captado diferentes características, pudiera mantener algo de estabilidad.
   e) Se anotaron las medias de los resultados.
3. **Fidelidad:**
   a) Se ejecutó el algoritmo en las mismas imágenes utilizadas para las pruebas de estabilidad.
   b) Cada imagen se procesó 5 veces, obteniendo la media en cada porcentaje.
   c) En el caso de algunos algoritmos, también se realizaron pruebas sobre sus capacidades en sus diferentes capas, lo cual se detallará más adelante.

En algunos resultados, se reportará la técnica utilizada con un número siguiéndolo (por ejemplo: LIME_100). En el caso de LIME y Deep SHAP, se trata del número de samples que se realizaron sobre la imagen. En el caso de kernel SHAP, también se trata del número de imágenes insertadas en el dataset de background que se le debe entregar (como en la explicabilidad por cohortes).
Estos números son diferentes para cada técnica, y se hizo de tal manera, que se intentase conseguir la mejor explicación posible en un tiempo razonable para cada algoritmo, por ello no se estandarizó. No tiene sentido intentar correr Deep SHAP con 1000 samples como se podría hacer con LIME debido a que no mejorarían los resultados de manera significativa, especialmente si tenemos en cuenta que necesitaría muchos más recursos.
Es importante tener en cuenta que estos resultados son específicos para este estudio y no pueden ser generalizados o sacados de contexto sobre otros datasets distintos. Los números presentados están influenciados por el contexto y el background específico de la muestra estudiada. Por ende, los resultados solo se usarán de manera comparativa dentro del contexto de que todos los algoritmos han sido utilizados en el mismo ambiente.

5.1. Resultados modelo mascarillas

Recursos utilizados

Técnicas XAI	**Recursos Utilizados**			
	Memoria Gb	**Tiempo de CPU(s)**	**Function Calls**	**Primitive Calls**
LIME_100	9.42	1.029	75,624.70	75,101.70
SHAP Kernel 50	16.77	110.49	126,441,641	119,061,037
SHAP Deep 50	13.82	7.56	859,344.40	847,126.40
Grad-CAM	11.77	0.06	28,835	28,233
IG	9.15	1.09	31,723	31,373

Cuadro 3: Comparación de la métrica de recursos usando distintas técnicas XAI empleando el modelo de mascarillas.

En la tabla anterior se muestran los resultados para el modelo de las máscaras, con la tabla completa viéndose en el apéndice (Cuadro 11)

En este caso podemos empezar a apreciar la superioridad computacional de los modelos integrados, con Grad Cam siendo claramente el mejor computacionalmente, ocupando solo algo más de memoria que LIME. LIME muestra ser mejor que SHAP en cuanto a coste computacional y Deep SHAP es mucho más rápido que kernel SHAP como sería de esperar (explicaciones de las diferencias Deep SHAP:Definicion 9.4, y Kernel SHAP: Definicion 9.3)

Se pudo testear los recursos de kernel 150, pero a partir de ahora no se incluirán sus resultados, debido a su excesivo uso de memoria, que causaba la muerte del kernel en el resto de tests.
LIME y ambos SHAP pueden variar dependiendo de los números de samples utilizados, que se ven en los resultados completos.

Estabilidad

Técnicas XAI	Estabilidad	
	Cosine Similarity	**Pearson Correlation**
LIME100	0.946	0.784
LIME500	0.944	0.778
LIME1000	0.935	0.741
Deep SHAP50	0.534	0.534
Deep SHAP100	0.637	0.637
Deep SHAP200	0.560	0.560
Kernel SHAP 50	0.00	-0.00
Kernel SHAP 100	0.00	-0.00
Grad Cam	0.939	0.926
IG	0.783	0.783

Cuadro 4: Métrica de estabilidad de las diferentes técnicas de XAI sobre el algoritmo de máscaras.

En la tabla anterior, se ven resultados que muestran la aleatoriedad de LIME y Deep SHAP, con LIME en este caso acertando de forma casi perfecta en el plano vectorial del resultado, lo que muestra que se fija bien en las secciones, en cuanto a orden de importancia, en cambio, el coeficiente de correlación de Pearson nos enseña que LIME al tener aleatoriedad, no están tan relacionadas las diferentes explicaciones, lo que nos indica que sus magnitudes son diferentes entre cada iteración, por lo que las secciones menos relevantes (que tienen menos magnitud) de la explicación, varían mucho.

Con Deep SHAP, se ve que aunque encuentre similitudes entre las explicaciones, sigue habiendo diferencias. Lo que nos dicen ambas métricas es que encuentra una similitud entre las explicacionesde secciones son importantes y sus magnitudes y variación entre ellas, pero aunque existan, sigue existiendo bastante variabilidad entre explicaciones.

Kernel SHAP, en cambio, muestra resultados bastante chocantes, siendo ortogonal de media en Cosine, y mostrando casi 0 dependencias entre las explicaciones, aunque si las explicaciones ya eran tan diferentes de primeras con cosine, era de esperar que no tuviesen casi ninguna relación entre ellas. Esto nos dice que el modelo no está encontrando casi nada en común entre explicaciones.

Desde ahí, se ve que el método intrínseco Grad Cam es el más estable con diferencia, lo cual tiene sentido, al tener acceso directo a los weights y valores durante la ejecución del proceso.

IG, en cambio, muestra mucha menos estabilidad. Se debe a la integracion a traves de las capas. Se muestra que se fija relativamente bien en ambas métricas, aunque para ambas se ve que sigue existiendo variabilidad dentro de las explicaciones. Entonces existe variabilidad dentro de las secciones a las que se atribuye importancia, y las magnitudes relativas a estas.

Fidelidad

Técnicas XAI	Fidelidad	
	AUC Inserción	AUC Deleción
LIME_100	89.92	9.94
LIME_500	88.57	18.91
LIME_1000	95.76	17.05
Deep_SHAP_50	51.72	40.30
Deep_SHAP_100	51.73	31.99
Deep_SHAP_200	52.29	39.95
Kernel_SHAP_50	11.46	73.21
Kernel_SHAP_100	11.47	73.16
Grad_CAM_avg	85.36	62.76
IG	60.35	56.69

Cuadro 5: Comparación de Fidelidad de técnicas XAI sobre el modelo de mascarillas. Los resultados están expresados en porcentajes.

Aquí se muestran solo los resultados, los gráficos se muestran figura 24
Se ve como LIME mejora cuantos más samples tiene, aunque incremente su deleción, su inserción mejora considerablemente. SHAP muestra resultados sorprendentemente bajos en inserción y altos en deleción (comportamiento opuesto a lo esperado en una buena métrica de fidelidad) en la variante Deep, y sub pares con kernel, mostrando que aunque en teoría kernel es agnóstico y puede manejar cualquier dato con tal de que se introduzca la información en su formato, está hecho para tratar con datos tabulares, lo que muestra que no tiene la capacidad de realizar análisis de datos excesivamente complejos.

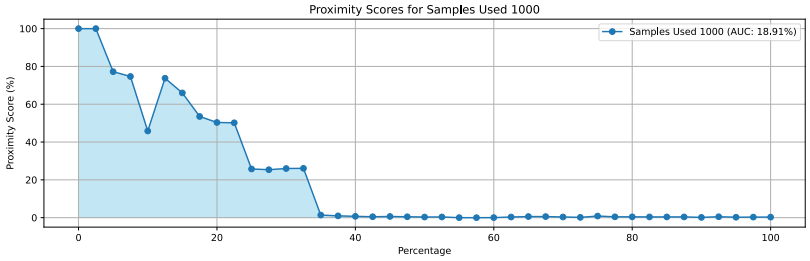


Figura 21: Gráfica de deleción LIME.

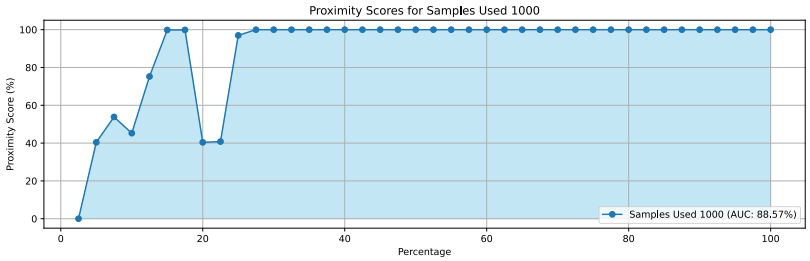


Figura 22: Gráfica de inserción LIME.

En las imágenes , puede verse como se representan las métricas en los grafos, con el resto viéndose en figura 24. Para todos los algoritmos, cuanto mayor sea el área para inserción y menor para deleción, mejor técnica de explicabilidad se considerará desde el punto de vista de fidelidad.

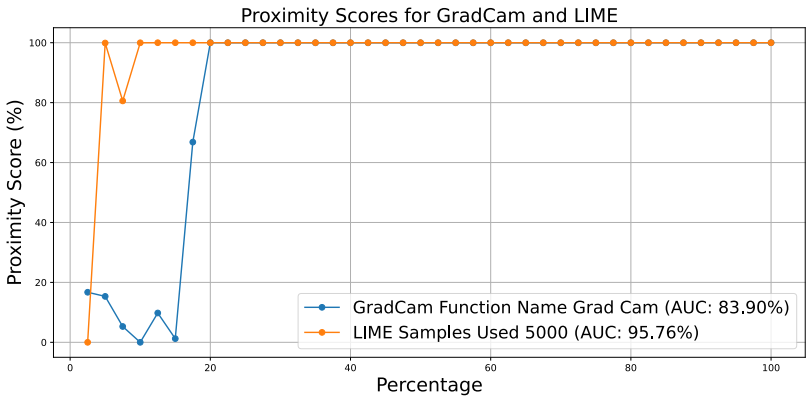


Figura 23: Gráfica de deleción LIME vs Grad Cam.

En la imagen anterior se muestra una comparación gráfica entre dos de las dos técnicas (100 samples de LIME vs GradCam), donde puede observarse que LIME encuentra rápidamente la solución, mientras que Grad Cam necesita más porcentaje de píxeles de la imagen para llegar a un 100 % de la explicación (alrededor de un 20 % de los píxeles más relevantes).

5.2. Resultados modelo AutoEncoder

En primer lugar, no se podrán comparar los resultados obtenidos mediante Kernel SHAP, dado que este explicador requiere que las perturbaciones proporcionen suficiente información para una de sus bibliotecas base (LASSO, el proceso de IRC en particular), lo cual no está garantizado en este modelo sin un número considerable de imágenes de background, en particular, al menos 150. El problema radica en que este número no es suficiente y causa fallos recurrentes en el kernel, por lo que se optará por utilizar únicamente los resultados del modelo de máscara (Mask model) para SHAP.

Además, Deep SHAP no es compatible con la estructura del modelo, al estar hecho con tensorflow en mente, no puede adaptarse a la estructura de un autoencoder de torch, lo que imposibilita su implementación.

Todos los puntos comentados anteriormente, arrojan a la luz el hecho de que en el caso de estar interesados por una única técnica empleada para la explicabilidad, las técnicas anteriores deberían descartarse, al tener muchas particularidades que impiden su uso sobre cualquier tipo de modelo.

Técnicas XAI	**Recursos Utilizados**			
	Memoria Gb	**Tiempo de CPU(s)**	**Function Calls**	**Primitive Calls**
LIME_100	0.76	4.772	74,093	63,038
Grad_cam	1.29	0.144445	2517.5	2268.5
IG	1.29	6.52857	126,613	114,211

Cuadro 6: Comparación de los distintos recuros de Técnicas XAI aplicadas sobre el modelo AutoEncoder.

Aquí se muestran los resultados preliminares, la tabla completa se encuentra en él Cuadro 12

Se vuelve a ver la gran diferencia de uso de recursos entre los diferentes modelos, con aquellos algoritmos especializados para el tipo de modelo autoencoder (IG y Grad), siendo claros ganadores

en cuanto a tiempo requerido. Grad Cam en específico es superior con creces en cuanto a rendimiento con el menor tiempo, y muy poca memoria requerida. La representación de LIME ha obtenido buenos resultados, siendo la mas basica. Se pueden ver el resto de resultados en la tabla completa. Como en LIME pueden cambiarse el número de samples, los recursos varian dependiendo de este número.

Estabilidad

Técnicas XAI	**Estabilidad**	
	Cosine Similarity	**Pearson Correlation**
LIME_100	0.508	0.500
LIME_500	0.512	0.504
LIME_1000	0.509	0.504
Grad_Cam	0.828	0.923
IG	0.997	0.919

Cuadro 7: Estabilidad del modelo AutoEncoder.

Los resultados son muy indicativos del tipo de explicaciones que aporta cada uno de los algoritmos, con LIME siendo un método con aleatoriedad, causando grandes diferencias entre explicaciones. Se ven buenos resultados de Grad Cam e IG con bastante poco cambio entre explicaciones, con ese cambio siendo más debido al cambio mínimo entre imágenes que a cambios de resultado en el modelo, al contrario de lo que parecería indicar LIME.
También se puede apreciar una gran diferencia entre los resultados de este test con el anterior, en especial para LIME, y se debe en su mayor parte al tipo de pruebas que realiza LIME.
Al hacer un análisis de perturbación sobre un modelo tan complicado como este, le es increíblemente fácil asumir que diferentes secciones pueden tener importancias diferentes.
Este problema ya se mencionó en el análisis del coeficiente de correlación de Pearson, viéndose mucho menor incluso en el anterior test, en cambio, aquí también le es difícil captar el vector representado por el cosine similarity, pero tiene el problema de no poder fijarse bien en este caso, en la importancia general de cada sección, debido a que al modelo de anomalías, cualquier cambio significativo e incluso menor le causara convertirse en una anomalía casi siempre.

Con un número limitado de samples, debe intentar hacer una aproximación en vez de una prueba muy exhaustiva que tardaría mucho más. Esto supone que se fija bien en las zonas de importancia, pero no puede captar bien ni sus magnitudes, al igual que en el anterior test, ni la dirección general, al contrario que en el anterior test, debido a la mayor complejidad, como se verá en el apartado de fidelidad.

IG y Grad Cam tienen estabilidades prácticamente perfectas, con un poco de variación mostrada en el cosine de Grad Cam, pero casi insignificante, teniendo en cuenta que puede causarse mínimamente por los posibles cambios debido al ruido. Más raro es lo poco que cambia IG con un 0.997, que puede a llegar a ser peor que no tener algo de variación, porque implica que no está captando correctamente ningún cambio, excepto en magnitudes, como se puede ver en el Pearson correlation.

Fidelidad

Técnicas XAI	Fidelidad	
	AUC Inserción	AUC Deleción
LIME_100	95.51	5.81
LIME_500	95.53	5.07
LIME_1000	95.54	5.09
Grad_CAM_avg	45.20	61.53
Grad_Cam_Best	92.13	25.34
Grad_Cam_Del	32.96	8.59
IG	16.91	40.12

Cuadro 8: Comparación de Fidelidad de Técnicas XAI autoencoder mostrado en porcentajes.

Aquí se muestran solo los resultados,los gráficos se encuentran en la figura 34

Se ve muy claramente que aunque no sean muy estables las explicaciones de LIME, siguen siendo muy fiables, con el algoritmo fijándose siempre bien en lo importante, solo que no es numéricamente estable en cuanto a estos resultados.

En cuanto a fidelidad, Grad Cam se trata de un algoritmo de gradientes, por lo que si se fija en una zona, se verá que todo el alrededor queda resaltado, causando que los valores más grandes se concentren alrededor de puntos específicos, que aun siendo buena información, causan que en este caso, con el modelo de anomalías, se centre en demasiada información que no cambia el resultado. En algunas capas sí tiene buen AUC, pero de media, se ven resultados similares a la gráfica mostrada.

Grad Cam tiene resultados muy variados entre las diferentes capas, con la capa 7 teniendo AUC de 92 para inserción y 25.34 de deleción, con la capa 23 teniendo AUC de 8.59 en deleción, pero 32.96 en inserción. La diferente variación de valores entre capas, muestra que como sé preveía, cada una se fija en diferentes secciones, entonces algunas capas van a tener disparidades muy grandes entre ellas en cuanto a la importancia de secciones de las imágenes.

Le pasa algo muy similar a IG, aunque el efecto es menos pronunciado, fijándose en aquellas partes del modelo más relevantes para las neuronas, pero también atribuye importancia a los alrededores de estas secciones, aunque menos fuertemente que Grad Cam, causando que le asigne importancia a partes que no tienen verdadera relevancia en este modelo.

6. Conclusión

Gracias al presente estudio de métricas sobre las diferentes herramientas de explicabilidad aplicadas sobre los modelos presentados, puede observarse que la decisión sobre qué técnica de explicabilidad utilizar, dependerá de la implementación y de los recursos necesarios.

En caso de querer la explicación más fiable, LIME se adapta muy bien a los modelos, aunque la estabilidad deja mucho que desear en modelos más complejos (Definicion 9.5.1). LIME consigue la explicación más fiable numéricamente.

Si nos fijamos en algunos otros tests que se hicieron, visto específicamente en (Definicion 9.5.1) también se puede decir que la estabilidad de LIME depende de cuan complejos son el modelo y dataset.

SHAP muestra que no es especialmente bueno para modelos de visión por computadora. Puede que en algunos casos particulares pueda llegar a tener buenos resultados, pero ni la implementación de kernel ni Deep buenos resultados en ninguno de los aspectos.

Grad Cam es una buena alternativa a LIME, utilizando muchísimos menos recursos, manteniendo una estabilidad casi perfecta, y llegando a buenos resultados en fidelidad, aunque inferiores a LIME.

Tiene la particularidad de que es necesario saber escoger la capa correcta, siendo esto imprescindible, debido a la gran diferencia entre resultados que puede llegar a dar.

Finalmente, IG es un algoritmo con resultados mediocres, no tiene resultados buenos en cuanto a fidelidad y tiene peor uso de recursos que Grad Cam. Tiene buenos resultados en cuanto a estabilidad.

Por lo tanto, en caso de querer la máxima precisión del algoritmo, LIME es la elección. Sin embargo, si se necesitase realizar muchas de estas explicaciones, como puede llegar a ser el caso con explicaciones en tiempo real, podría llegar a ser mejor utilizar Grad Cam, al ser explicaciones estables, que gastan menos recursos y así se reduce el impacto medioambiental, aun manteniendo una buena precisión de explicación.

También se puede ver, que las explicaciones no tienen por qué tener cambios enormes en modelos de anomalías, con la inserción y deleción siendo muy poco útiles en la evaluación de las herramientas, debido a como toma decisiones el modelo.

Finalmente, hay que remarcar que las métricas también pueden variar de implementación a implementación, especialmente teniendo en cuenta las diferentes librerías, arquitecturas de modelos, y complejidad de la base de datos. Estos resultados solo deberían ser una base sobre la que seguir trabajando.

7. Evaluación

Recursos Utilizados: Debido a la naturaleza de estos tests, es casi seguro que existan maneras más eficientes de implementar varias de las librerías, ya sea con menos llamadas, o implementaciones con mejor acceso a la arquitectura del modelo. Los recursos pueden variar bastante de implementación a implementación, pero en general, no deberían poder cambiar tanto como para cambiar los rankings mostrados. Por ejemplo, SHAP kernel, nunca debería poder ser más rápido que LIME.

Estabilidad: Debido a la naturaleza de LIME y SHAP, sus estabilidades son más volátiles, y en caso de tener más iteraciones podrían fluctuar. Sería mejor tener mas en cuenta su carácter aleatorio, que el valor numérico, y simplemente utilizarlo de guía. Para mejorar esta sección, lo mejor hubiese sido generar un pequeño dataset con las imágenes perturbadas antes de tiempo, y usar siempre estas mismas para los tests. Esto hubiese garantizado igualdad absoluta ante posibles diferencias de estabilidad, aunque el gran número de datos, debería haber bajado significantemente este posible error.

Fidelidad: El mayor problema de Grad Cam e IG es como se fijan en los puntos de importancia, asignando importancia a píxeles colindantes, por lo que es difícil saber si es justa la asignación de valor.

También hubiese sido mejor encontrar una función de normalización, que pudiese representar mejor los casos de IG y Grad Cam para el modelo de máscaras, ya que la distancia máxima no es aquella que tiene con el resultado final. Esto es especialmente importante para la deleción de IG, aunque sus resultados no hubiesen mejorado.

Por último, en caso de tener más tiempo, también se podrían evaluar muchas otras técnicas, como puede ser el caso de Occlusion sensitivity, método de atención basado en un modelo de transformadores, o RISE(Randomized Input Sampling for Explanation).

8. Trabajo futuro

En un campo en constante evolución, pueden surgir cambios de manera repentina y desarrollos de nuevas estructuras de modelos, lo que requiere el estudio de nuevas técnicas y la adaptación de métricas, conforme se establezcan estándares en el sector.

Será necesario desarrollar métricas para futuras técnicas aplicables a modelos aún no diseñados. Habra que buscar métricas agnósticas y métricas más subjetivas, con el objetivo de democratizar su uso.

Será necesario cuantificar comó se defiende cada modelo ante ataques adversariales. Habrá que estudiar como se comportan los algoritmos y métodos ante los ataques adversariales, y ver como reducen su impacto.

El objetivo principal es hallar una explicación interpretable, minimizando el uso de recursos, ya que aumentara el uso de IA y modelos cada vez más complejos. Si las explicaciones se requieren por ley, se buscará reducir el impacto ambiental asociado a estas.

Referencias

[1] European Parliament. Proposal for a regulation of the european parliament and of the council on artificial intelligence act. European Parliament document, 2024.

[2] United States Congress. Artificial intelligence accountability act of 2024. United States Congressional Record, 2024.

[3] Christoph Molnar. Interpretable machine learning. a guide for making black box models explainable, 2019.

[4] Censius AI. What is global, cohort and local explainability?, 2021. Accessed: 2024-05-27.

[5] Arize AI. What are global, cohort and local model explainability?, 2021. Accessed: 2024-05-27.

[6] Qlik. Tutorial - reviewing versions, 2024. Accessed: 2024-06-20.

[7] Zeren Tan, Yang Tian, and Jian Li. Glime: General, stable and local lime explanation. Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), 2023.

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[9] Ian Covert. Explaining machine learning models with shap and sage, 2020. Accessed: 2024-05-27.

[10] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[11] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017.

[12] Hyun-Lim Yang, Jong Kim, Jong Kim, Yong Koo Kang, Dong Ho Park, Han Sang Park, Hong Kim, and Minsoo Kim. Weakly supervised lesion localization for age-related macular degeneration detection using optical coherence tomography images. PLOS ONE, 14:e0215076, 04 2019.

[13] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization, 2016.

[14] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks, 2018.

[15] Yipei Wang and Xiaoqian Wang. Benchmarking deletion metrics with the principled explanations, 2024.

[16] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation, 2017.

[17] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models, 2018.

[18] Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Zhiwei Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations, 2021.

[19] Avi Rosenfeld. Better metrics for evaluating explainable artificial intelligence. 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), 2021.

[20] Alexandr Oblizanov, Natalya Shevskaya, Anatoliy Kazak, Marina Rudenko, and Anna Dorofeeva. Evaluation metrics research for explainable artificial intelligence global methods using synthetic data. 2023.

[21] Lars Kai Hansen Laura Rieger. A simple defense against adversarial attacks on heatmap explanations, 2020.

[22] Yifei Zhang, Siyi Gu, James Song, Bo Pan, and Liang Zhao. Xai benchmark for visual explanation, 2023.

[23] Laura Rieger and Lars Kai Hansen. A simple defense against adversarial attacks on heatmap explanations. arXiv preprint arXiv:2007.06381, 2020.

[24] Python Software Foundation. cprofile: A deterministic profiler for python programs, 2023.

[25] Fabian Pedregosa. memory_profiler: Monitor memory usage of a python program, 2023.

[26] Serg Masís. Interpretable machine learning with python. https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python, 2021. Accessed: 2024/04/09.

8.1. Referencias adicionales

Estas referencias fueron también revisadas durante el trabajo, pero no fueron las principales fuentes de información, por lo que no incluyen un número ni son referenciadas en el texto.

- Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In International Conference on Machine Learning, pp. 2673–2682, 2018.
- Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., and Zhu, J. Defense against adversarial attacks using high-level representation guided denoiser. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1778–1787, 2018.
- Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems, pp. 4765–4774, 2017.
- Mohseni, S. and Ragan, E. D. A human-grounded evaluation benchmark for local explanations of machine learning. arXiv preprint arXiv:1801.05075, 2018.
- Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition, 65:211–222, 2017.
- Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. Improving adversarial robustness via promoting ensemble diversity. arXiv preprint arXiv:1901.08846, 2019.
- Rieger, L., Chormai, P., Montavon, G., Hansen, L. K., and Müller, K.-R. Structuring Neural Networks for More Explainable Predictions. pp. 115–131. Springer, Cham, 2018.
- Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to Compose Neural Networks for Question Answering. In *The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1545–1554, 2016.
- Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2956–2964, 2015.

- Mark W Craven and Jude W Shavlik. Extracting Comprehensible Models from Trained Neural Networks. PhD thesis, University of Wisconsin, Madison, 1996.
- Piotr Dabkowski and Yarin Gal. Real Time Image Saliency for Black Box Classifiers. In *Neural Information Processing Systems*, pages 6970–6979, 2017.
- Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.
- Adversarial attacks and defenses in explainable artificial intelligence: A survey, by Hubert Baniecki and Przemyslaw Biecek *https://ar5iv.labs.arxiv.org/html/2306.06123*
- Calibrated Explanations: with Uncertainty Information and Counterfactuals
- Attention is not Explanation Sarthak Jain, Byron C. Wallace *https://arxiv.org/abs/1902.10186*
- Attention is not not Explanation Sarah Wiegreffe, Yuval Pinter *https://arxiv.org/abs/1908.04626*

9. Apéndice

9.1. Autoencoders

Los autoencoders son una clase de redes neuronales utilizadas principalmente en el aprendizaje no supervisado, para aprender representaciones eficientes de los datos de entrada. La idea básica detrás de un autoencoder es que aprende a comprimir la información de entrada en una representación de dimensionalidad menor, y luego a reconstruir la entrada original a partir de esta representación comprimida.

Componentes de un Autoencoder

1. **Codificador (Encoder):** El codificador toma la entrada y la transforma en una representación de dimensionalidad más baja. Esta representación comprimida contiene la información más relevante de los datos de entrada.
2. **Decodificador (Decoder):** El decodificador recibe la representación comprimida del codificador y trata de reconstruir la entrada original a partir de ella. El objetivo es que la reconstrucción sea lo más fiel posible a la entrada original.

Funcionamiento del Autoencoder

1. **Codificación:** La entrada pasa a través del codificador, que aplica transformaciones lineales y funciones de activación para reducir la dimensionalidad de los datos y obtener una representación comprimida.
2. **Decodificación:** La representación comprimida se pasa al decodificador, que intenta reconstruir la entrada original a partir de ella. El decodificador aplica transformaciones inversas a las del codificador para obtener la reconstrucción.
3. **Función de Pérdida:** Se utiliza una función de pérdida para comparar la entrada original con la salida del decodificador. Esta función de pérdida cuantifica la discrepancia entre la entrada y la reconstrucción, incentivando al modelo a aprender una representación significativa de los datos.

4. **Entrenamiento:** Durante el entrenamiento, los pesos de la red se ajustan para minimizar la función de pérdida, lo que implica que la reconstrucción se ajusta cada vez más a la entrada original.

9.2. BCE

La **Entropía Cruzada Binaria** (BCE, por sus siglas en inglés) es una función de pérdida comúnmente utilizada en problemas de clasificación binaria en el campo del aprendizaje automático. Esta función mide el rendimiento de un modelo cuyas salidas son probabilidades en el rango de 0 a 1. La BCE compara la salida predicha del modelo con el valor verdadero esperado, que debe ser 0 o 1. A continuación se describen los componentes de la BCE:

- p es la probabilidad predicha por el modelo de que la instancia $\mathbf{x}$ pertenece a la clase 1.
- y es la etiqueta real de la instancia $\mathbf{x}$, donde $y \in \{0, 1\}$.
- La función de pérdida BCE se define como:

$$L(y, p) = -\left(y \log(p) + (1 - y) \log(1 - p)\right)$$

- Esta función calcula el logaritmo de la probabilidad predicha p si la etiqueta real y es 1, y el logaritmo de $1 - p$ si la etiqueta real es 0, sumando estos términos de manera ponderada por las etiquetas reales.

9.3. Kernel SHAP

Kernel SHAP es un método interpretativo que aplica valores de SHAPley, provenientes de la teoría de juegos, para determinar la importancia de cada característica en las predicciones de un modelo de machine learning. Funciona de manera independiente al modelo, lo que significa que puede ser aplicado a cualquier tipo de modelo, desde regresiones lineales hasta modelos más complejos. Kernel SHAP evalúa la contribución de cada característica, simulando la inclusión y exclusión de estas en diferentes combinaciones, y observando el efecto en la predicción. Es el método más lento y básico de explicabilidad de SHAP, pero tiene la ventaja de ser completamente agnóstico, necesitando solo una función de predicción, y los datos. Este método está diseñado para datos tabulares, pero puede adaptarse a imágenes.

9.4. Deep SHAP

Deep SHAP es una técnica específica diseñada para explicar las contribuciones de las características en modelos de aprendizaje profundo. Basado en la metodología de DeepLIFT, Deep SHAP modifica este enfoque para incorporar los principios de los valores de SHAPley. Al descomponer las predicciones y analizar cómo las contribuciones de las características se propagan a través de las capas de una red neuronal, Deep SHAP ayuda a entender el rol de cada entrada en la decisión final del modelo.

9.5. Resultados modelo Anomalib

Técnicas XAI	**Recursos Utilizados**			
	Memoria(GB)	**Tiempo de CPU(s)**	**Function Calls**	**Primitive Calls**
LIME 100_5	1.3	300.69	1,107,057.9	965,982.9
LIME 100_10	1.3	300.52	1,107,059.9	965,984.9
LIME 1000_10	1.3	3004.60	10,917,351.2	9,507,276.2
LIME 1000_5	1.3	2992.29	10,917,352.3	9,507,277.3

Cuadro 9: Comparación de Técnicas XAI recursos Anomalib

Como se comento en capítulos anteriores, el modelo solo puede correr el explainer de LIME. Como es así, se comparara el rendimiento de este solo para este caso, y así vemos también si más samples garantiza que sea mejor el explicador. De primeras podemos ver que debido al modelo, ejecutar el explicador es increíblemente caro en cuanto a coste de computación. Después se verán costes más razonables, pero por ahora vemos que el número de labels asignados no influyen casi al tiempo de ejecución ni a la memoria. Desde ahí, el tiempo es casi igual a lo que se esperaría desde 100 a 1000, con alrededor de 10 veces más tiempo en correr.

Function	**Estabilidad**	
	Cosine Similarity	**Pearson Correlation**
LIME_1000_10	0.2906	0.1979
LIME_1000_5	0.2395	0.0440
LIME_100_10	0.3542	0.1762
LIME_100_5	0.3319	0.2071

Cuadro 10: Estabilidad Anomalib

Por la naturaleza aleatoria de LIME, los valores de estabilidad son bajos siempre, con diferencias muy grandes de explicación a explicación. Lo único que resalta en el test, es que LIME tiene casi ninguna consistencia, con la diferencia entre ellos siendo irrelevante, visualmente se puede ver que si se fija en algunos sitios, pero los valores asignados de explicación a explicación cambian demasiado, desde las regiones resaltadas, a los números asignados.
Los números presentados hacen referencia a la media, pero la variabilidad en la estabilidad en muy alta, con valores desde 0.8 a negativos, por lo que se necesitarían muchos más runs para asegurarse que estos valores son una buena representación de la media del explicador, pero tarda demasiado como para considerarse un test más extenso. Es por ello que solo se hizo la media igual que todo el resto de tests.

9.5.1. Conclusión sobre estabilidad vs complejidad en LIME

Lo más llamativo que se ve de este test es que en conjunto con el resto de tests del resto de modelos, se puede sacar una conclusión muy interesante, y es que LIME es menos estable cuanto más complicado es el modelo/dataset. La naturaleza aleatoria del algoritmo no mezcla bien con la complejidad. Viéndose como es una estabilidad muy buena en el clasificador, una estabilidad bastante más reducida con el autoencoder, y finalmente una estabilidad ridícula con Anomalib.

9.6. Modelo de mascarillas

Técnicas XAI	**Recursos Utilizados**			
	Memoria	**Tiempo de CPU(s)**	**Function Calls**	**Primitive Calls**
LIME_100	9.4	1.029	75,624.70	75,101.70
LIME_1000	9.4	8.356	668,585.40	663,832.40
LIME_5000	9.4	40.627	3,303,303.00	3,279,755.00
SHAP Kernel 50	16.8	110.49	126,441,641	119,061,037
SHAP Kernel 100	18.8	213.05	251,785,468	237,031,396
SHAP Kernel 150	24.4	318.61	377,173,779	355,043,166
SHAP Deep 50	13.8	7.56	859,344.40	847,126.40
SHAP Deep 100	14.0	13.76	859,426.30	846,208.30
SHAP Deep 200	16.3	26.40	865,078.40	851,782.40
Grad-CAM	11.8	0.058	28,835	28,233
IG	9.1	1.086	31,723	31,373

Cuadro 11: Comparación de Técnicas XAI recursos mask model

Aquí podemos ver como suben la necesidad de recursos según los samples para LIME, la memoria no sube debido a que la librería puede hacerlo en batches, pero tiempo y function calls si escalan con el número casi linealmente. SHAP kernel si necesita un escalado de memoria, debido al número de datos que necesita procesar a la vez en su background. Deep SHAP también tiene el problema de escalar su uso de memoria con el número de samples usados. El tiempo y funcion calls escalan casi linealmente en todos los casos.

Inserción

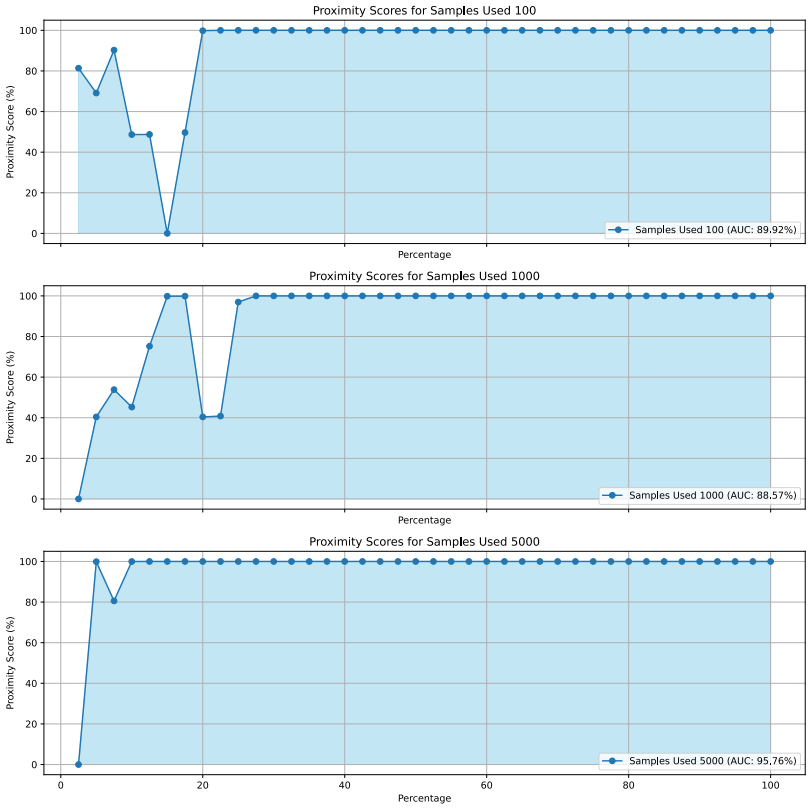


Figura 24: Las 3 gráficas de inserción LIME.

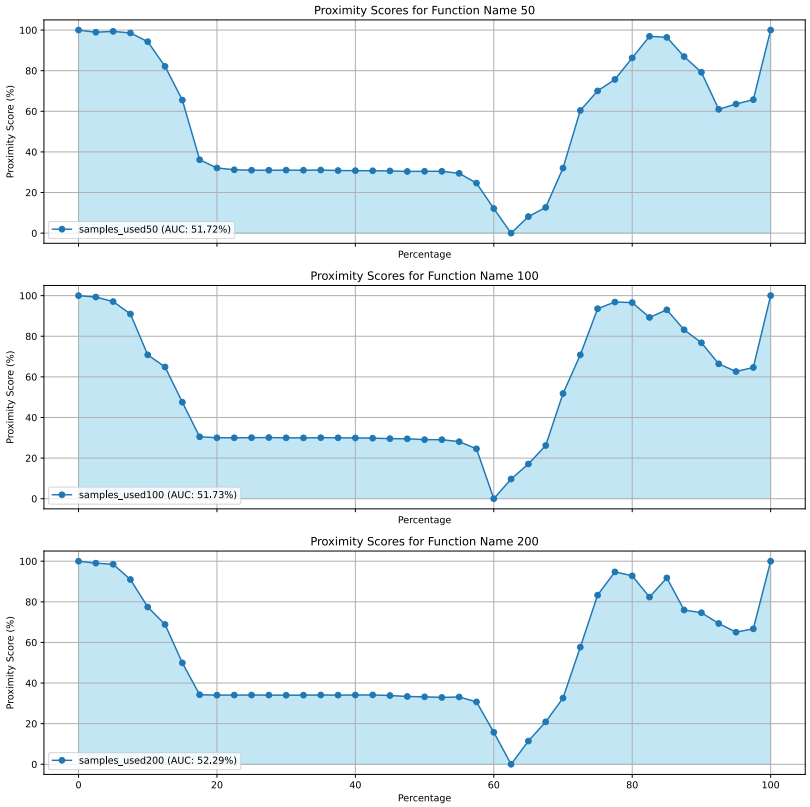


Figura 25: Gráfica inserción Deep SHAP.

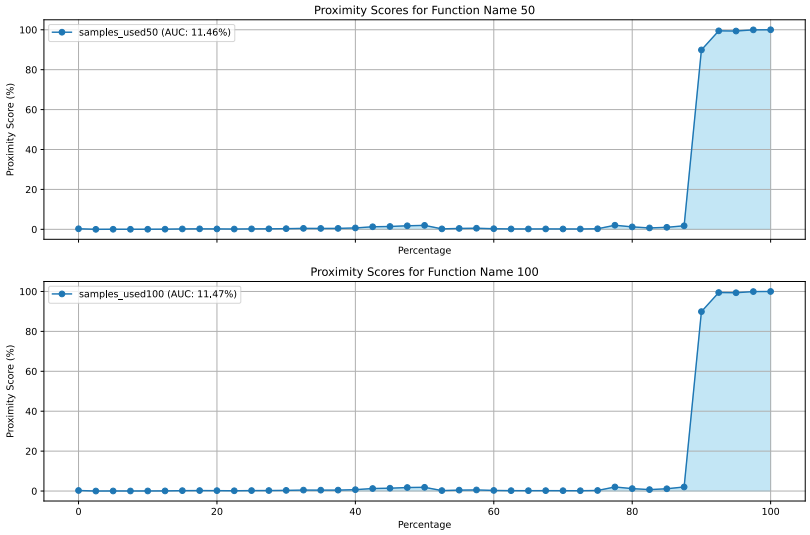


Figura 26: Gráfica inserción Kernel SHAP.

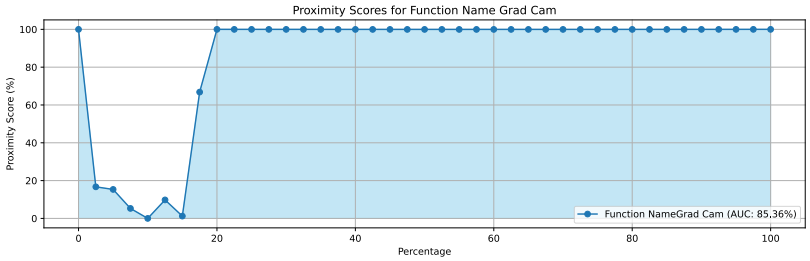


Figura 27: Gráfica de inserción de GradCam.

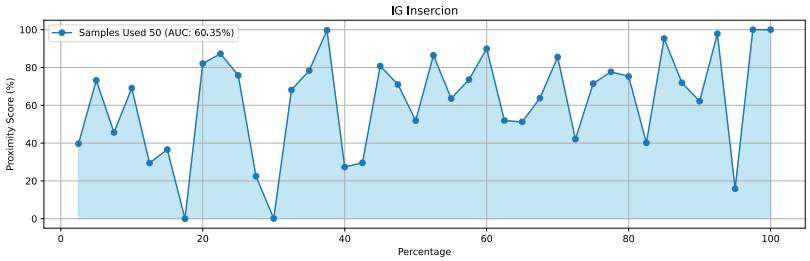


Figura 28: Gráfica de inserción IG.

Deleción

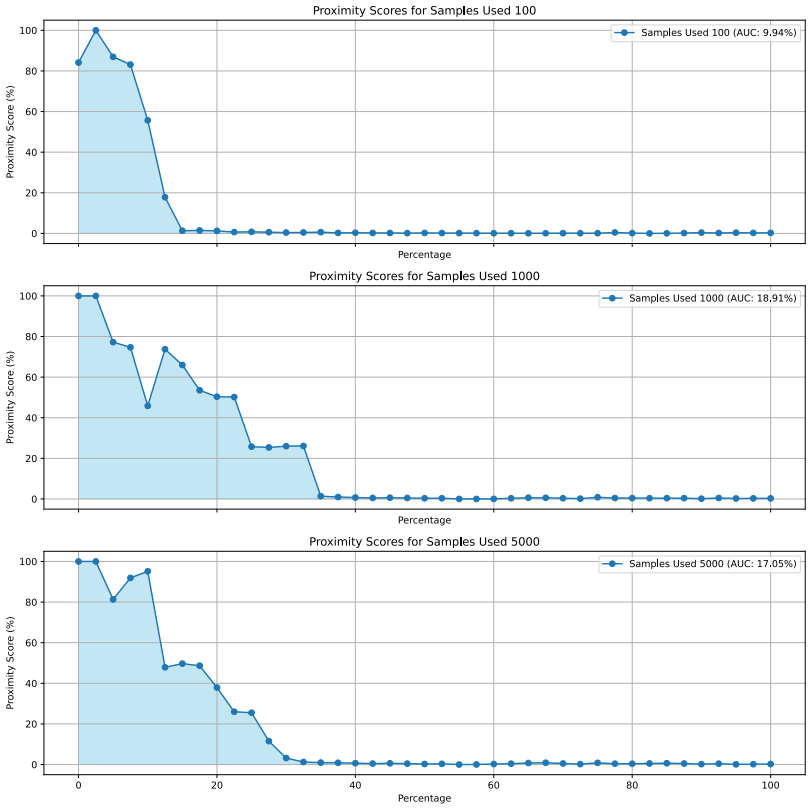


Figura 29: Las 3 gráficas de deleción LIME

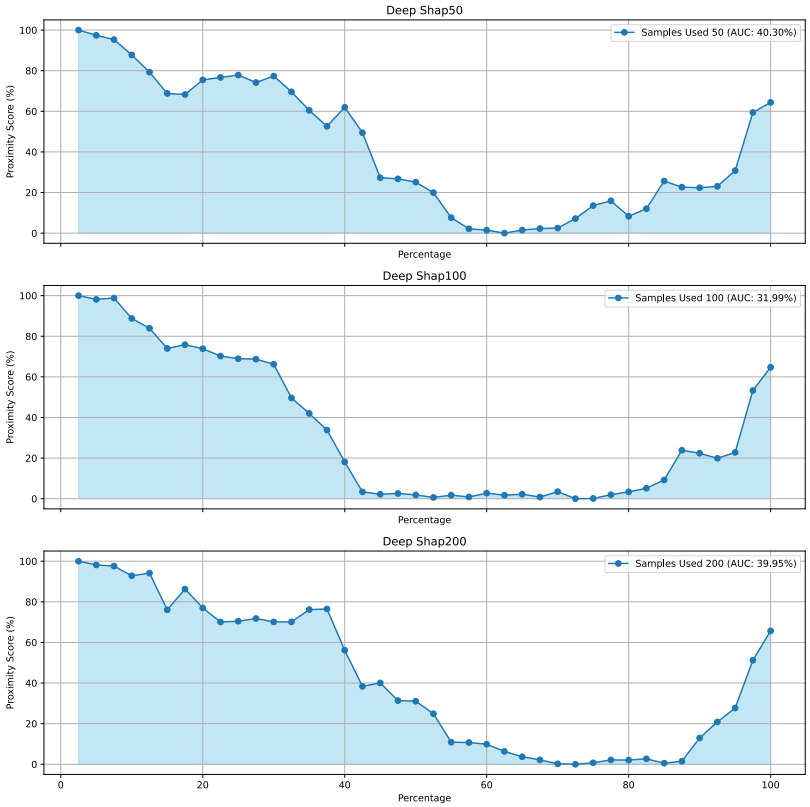


Figura 30: Gráfica de deleción Deep SHAP.

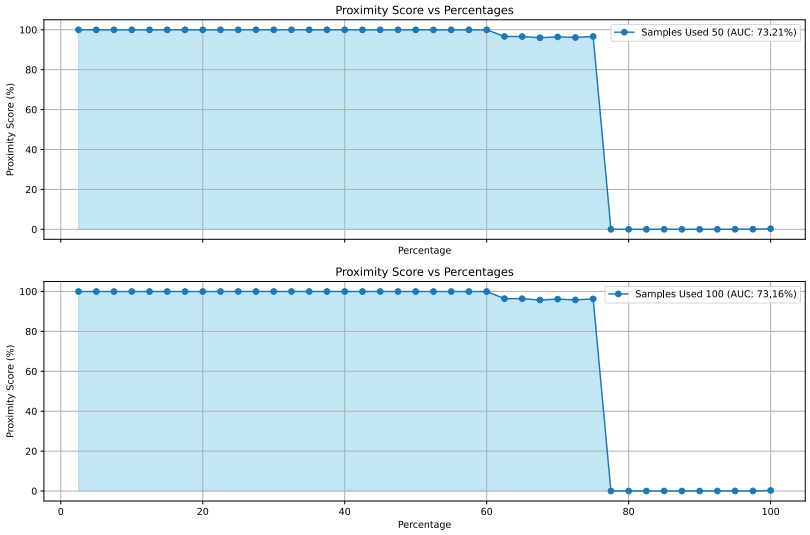


Figura 31: Gráfica de deleción Kernel SHAP.

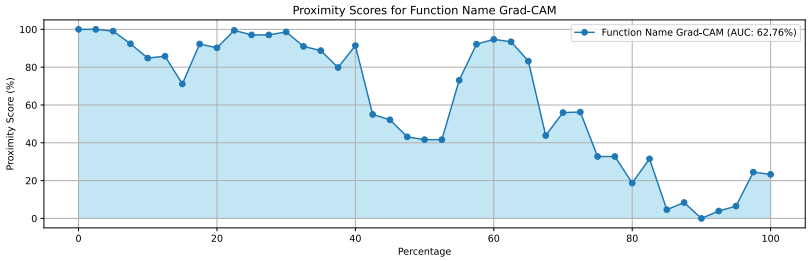


Figura 32: Gráfica de deleción GradCam.

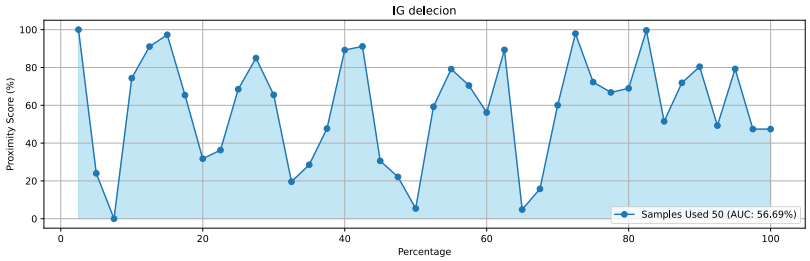


Figura 33: Gráfica de deleción IG.

9.7. Autoencoder

Técnicas XAI	**Recursos Utilizados**			
	Memoria	**Tiempo de CPU(s)**	**Function Calls**	**Primitive Calls**
LIME_100	0.78	4.772	74,093	63,038
LIME_500	0.76	22.941	344,170	289,075
LIME_1000	0.76	45.871	681,991	571,846
Grad_cam	1.29	0.144445	2517.5	2268.5
IG	1.30	6.52857	126613	114211

Cuadro 12: Comparación de Técnicas XAI recursos AutoEncoder updated

Aquí podemos ver como sube la necesidad de recursos según los samples para LIME, la memoria no sube debido a que la librería puede hacerlo en batches, pero tiempo y function calls si escalan con el número linealmente.

Inserción

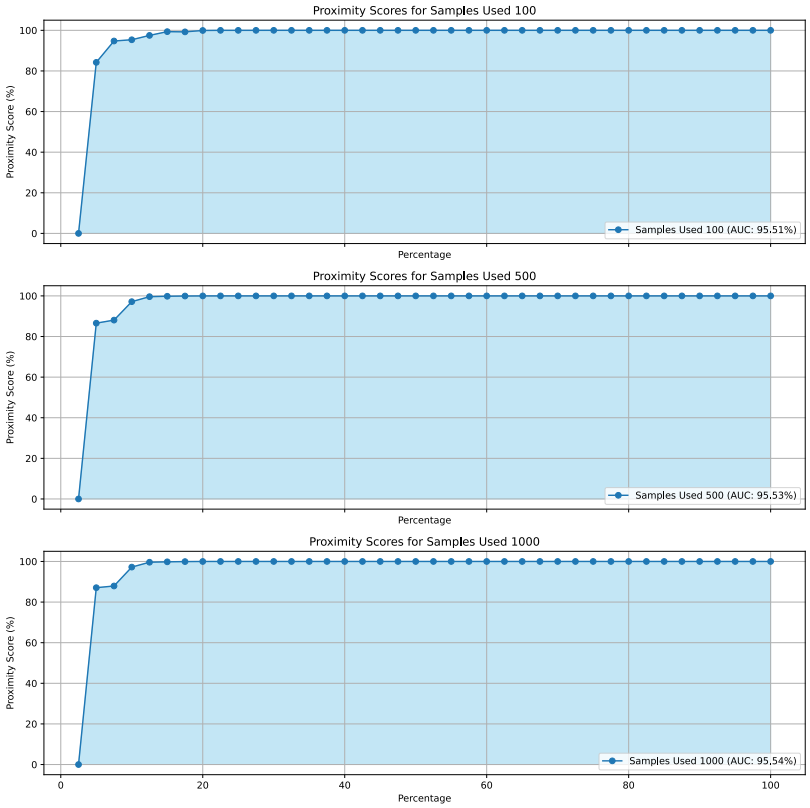


Figura 34: Las 3 gráficas de inserción LIME.

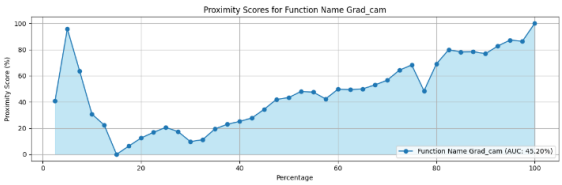


Figura 35: Gráfica de inserción GradCam.

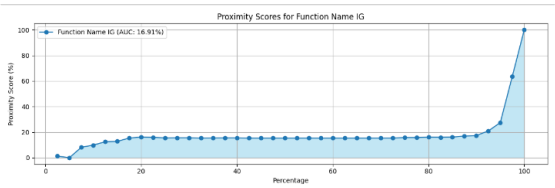


Figura 36: Gráfica de inserción IG.

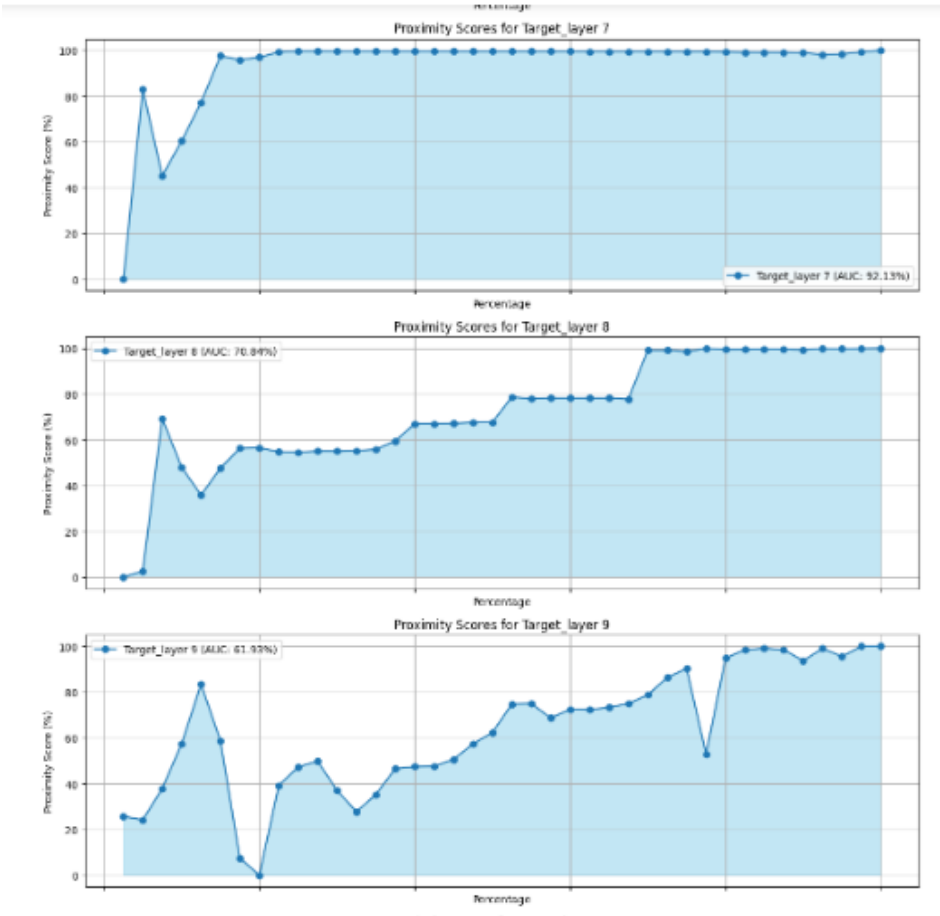


Figura 37: Las 3 gráficas de inserción de Grad Cam para las layers 7,8 y 9.

Deleción

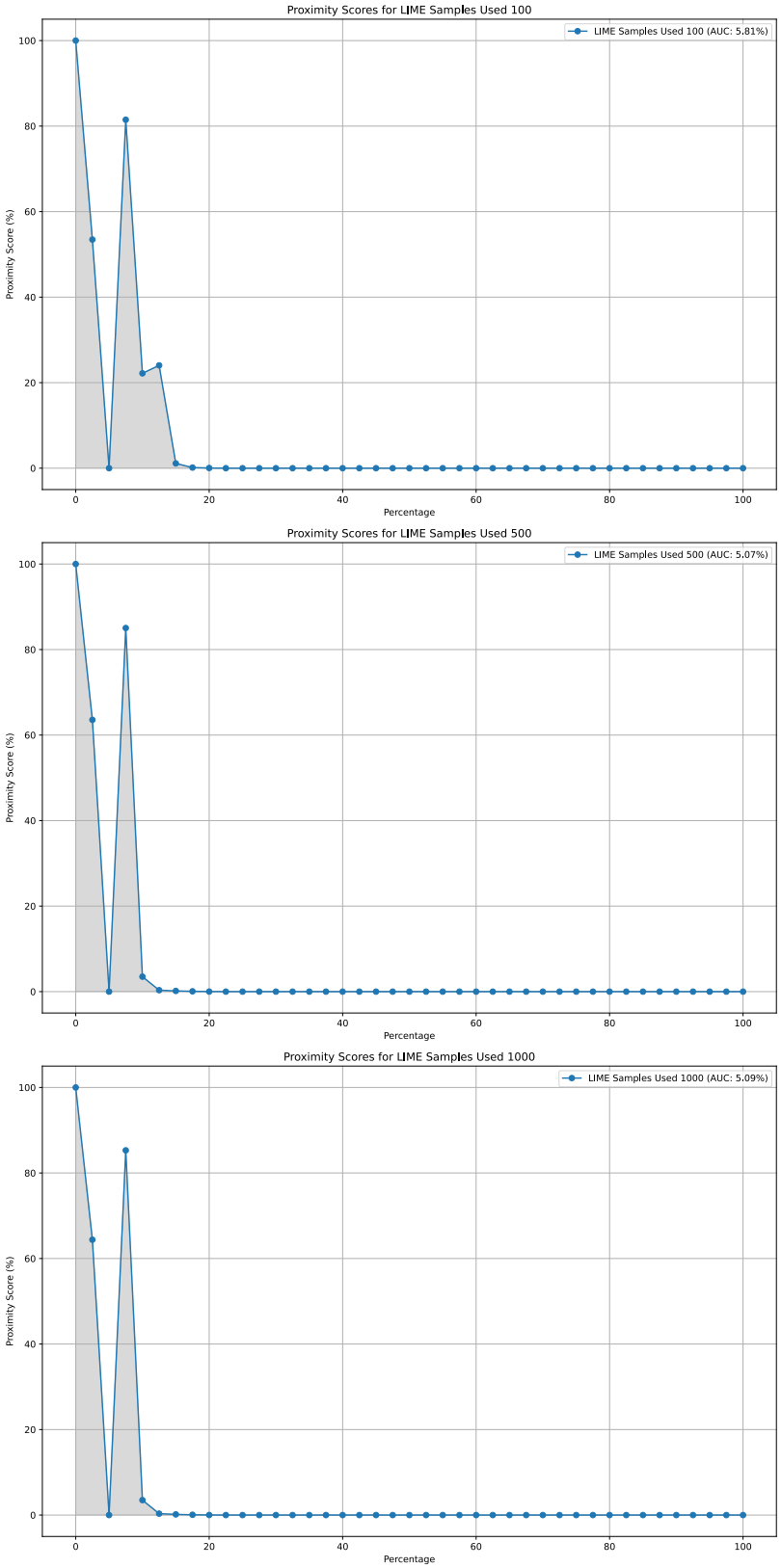


Figura 38: Las 3 gráficas de deleción LIME.

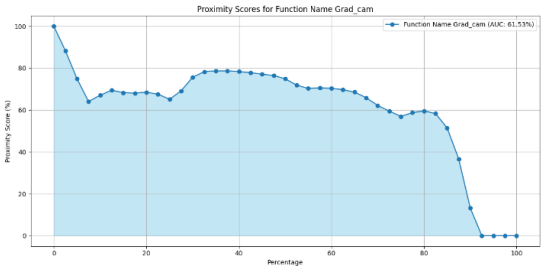


Figura 39: Gráfica de Grad_cam.

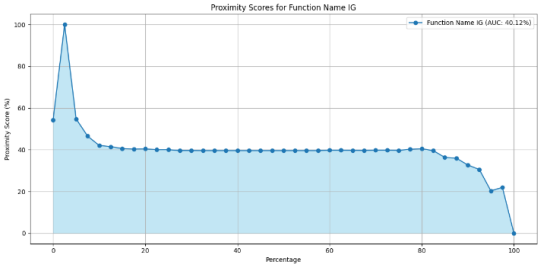


Figura 40: Gráfica de IG.