



UNIVERSIDAD DE GRANADA

DIAGNÓSTICO DE IMÁGENES HISTOLÓGICAS DE PRÓSTATA UTILIZANDO MÉTODOS DE EXPLICABILIDAD EN DEEP LEARNING

PEDRO GALLEGOS LÓPEZ

Trabajo Fin de Grado

Doble Grado en Ingeniería Informática y Matemáticas

Tutores

Francisco Herrera Triguero
Francisco Luque Sánchez

FACULTAD DE CIENCIAS

E.T.S. DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, a 20 de junio de 2022

DECLARACIÓN DE ORIGINALIDAD DEL TFG

Yo, **Pedro Gallego López**, con DNI 48261534J, Declaro que el presente Trabajo de Fin de Grado es original, no habiéndose utilizado fuentes sin ser citadas debidamente. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la Normativa de Evaluación y de Calificación de los estudiantes de la Universidad de Granada, de 20 de mayo de 2013, esto *conllevará automáticamente la calificación numérica de cero [...] independientemente del resto de las calificaciones que el estudiante hubiera obtenido. Esta consecuencia debe entenderse sin perjuicio de las responsabilidades disciplinarias en las que pudieran incurrir los estudiantes que plagien.*

Para que así conste lo firmo el 20 de junio de 2022

Fdo: Pedro Gallego López

Granada, 20 de junio de 2022

Yo, **Pedro Gallego López**, alumno del Doble Grado en Ingeniería Informática y Matemáticas de la **Facultad de Ciencias de la Universidad de Granada**, con DNI **48261534J**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Pedro Gallego López

Granada, 20 de junio de 2022

D. Francisco Herrera Triguero y D. Francisco Luque Sánchez, profesores del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Diagnóstico de imágenes histológicas de próstata utilizando métodos de explicabilidad en Deep Learning*, ha sido realizado bajo su supervisión por **Pedro Gallego López**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 20 de junio de 2022

Los directores:

Francisco Herrera Triguero

Francisco Luque Sánchez

Diagnóstico de imágenes histológicas de próstata utilizando métodos de explicabilidad en Deep Learning

Pedro Gallego López

Palabras clave:

Aprendizaje profundo, localización, CAM, explicabilidad, diagnóstico en imagen

Resumen

La explicabilidad surge dentro de la Inteligencia Artificial de la necesidad de entender los resultados que ofrecen los modelos de caja negra. Esta necesidad la tienen expertos del ámbito donde es aplicado el modelo. Por ejemplo en un modelo de diagnóstico de imágenes médicas, el experto sería el médico y necesitaría entender por qué el modelo da tales resultados. La necesidad también la tienen los desarrolladores del modelo para corregir posibles errores. En general, la explicabilidad es necesaria para la sociedad para poder poner medidas éticas en el uso de estos modelos.

En este trabajo se van a estudiar técnicas de explicabilidad aplicadas al diagnóstico de imágenes médicas (imágenes histológicas de próstata). Las técnicas de explicabilidad usadas son: Class Activation Map (CAM), Grad-CAM, Grad-CAM++ y Smooth Grad-CAM++. Estas técnicas tienen su base en CAM, que consiste en ver las características detectadas por el modelo para cada clase y en base a ello crear un mapa de activación que resalte aquellas zonas que han sido relevantes y han aportado para la decisión final del modelo. Cada una de estas técnicas incorpora una mejora a la anterior: Grad-CAM generaliza el concepto de CAM para más redes (quitando restricciones de arquitectura), Grad-CAM++ aporta una mejor ponderación para los mapas de activación ayudando a resaltar múltiples ocurrencias de una clase y Smooth Grad-CAM++ añade imágenes generadas a través de ruido añadido a la imagen original, y procesándolas, para tener un resultado más consistente. Estas técnicas se aplicarán sobre cuatro redes distintas: VGG16, RESNET18, MOBILENETV2 y EFFICIENTNETB0. De esta manera se conseguirá ver el desempeño de las técnicas (que deben ser independiente del modelo) de manera global entre todos los modelos, eliminando así de la evaluación de las técnicas el posible efecto que pueda tener cada modelo concreto usado en el problema.

El conjunto de datos es de imágenes histológicas de próstata a la que se tiene un acceso público: SICAPv1. Son imágenes de gran tamaño de 79 pacientes. Las imágenes han tenido que ser procesadas para disminuir su tamaño y, consecuentemente, provocar un aumento en el número de instancias del conjunto de datos que se va a usar.

El estudio consiste por un lado en hacer comparaciones de calidad de localización entre los modelos con cierta independencia de la técnica usada y por otro lado hacer

comparaciones entre las técnicas con cierta independencia de los modelos. De esta manera se podrá concluir un mejor modelo y una mejor técnica que en conjunto serán los seleccionados como mejor desempeño.

En este estudio se verá como la calidad de la clasificación influye en la calidad de la localización, y como modelos más clásicos y técnicas más básicas son las que mejor se comportan en este problema. Concretamente VGG16 será el mejor modelo junto a la técnica de Grad-CAM. Este estudio abre múltiples vías de estudio futuras.

Se dispone de un código de prueba de la herramienta dentro de Google Colab, al que se puede acceder desde el GitHub del proyecto:

<https://github.com/pedrogallegolpz/TFG>

Allí se podrán ejecutar más ejemplos que los mostrados en este trabajo.

Diagnosis of prostate histological images using Deep Learning explainability methods

Pedro Gallego López

Keywords:

Deep learning, localization, CAM, explainability, image diagnosis

Abstract

The boom of Artificial Intelligence in the world came a few years ago, when professions in all disciplines began to see its potential and tried to find applications in their sector. This explosion arrived with the mathematical models of Machine Learning. It was in fact the concept of machine learning that caused a revolution within humans themselves, breaking a fictional barrier created by humans many years ago. It was seen how Deep Learning models were able to solve very complex tasks with high performance. An example could be a Convolutional Network model solving a medical imaging diagnosis problem, where a human who is not skilled in the field would be unable to solve it. Such powerful results increased the confidence placed in this technology. Now, following the example, doctors could use this tool to diagnose, saving them effort and time.

However, Artificial Intelligence would have a barrier: explainability. Determining what decision to make when the doctor had a different opinion on the diagnosis than the deep learning model's opinion became a problem. The complexity of the models made it impossible to understand why they made those decisions, deeming them black boxes, completely opaque. This caused the doctor to make decisions solely on his own judgement, because if both the doctor and the model gave the same diagnosis then his judgement was followed; but if the doctor gave a diagnosis different from that of the model, the doctor was likely to follow his diagnosis because he understood it, assuming that the model could have been wrong in that case.

At this point XAI (eXplicable Artificial Intelligence) appeared whose mission was to provide Artificial Intelligence tools ensuring transparency in these models. This new field of study opened a wide variety of doors, such as ethics within Artificial Intelligence, where ethics can now be assessed within the decision field of the model; explainability for experts in the area of application of the model (such as doctors), allowing them to understand the results in order to help their own criteria; or explainability for experts in Artificial Intelligence (data scientists for example), detecting flaws in the model such as biases or other types of errors that may affect the goodness of the results.

Class Activation Map methods are tools that emerged from XAI in the area of Computer Vision. These methods or techniques allow you to understand what the model

has been based on to give the results it has produced. They visually create an activation map of the input, highlighting those regions that have been particularly relevant to the decision-making process. In the problem of diagnostic medical imaging, these techniques were able to determine which parts of the image were most relevant to the diagnosis. This meant that if the diagnosis was positive (pathological features), the techniques were able to determine the region where those features that marked the presence of pathology were found. So, these techniques were very powerful because it had achieved those models trained for classification were able to solve the task of unsupervised localization. Now, in this type of image-based diagnosis, a doctor can use these models judiciously, using them as a complement to his knowledge.

The object of study of this work is precisely the application of Class Activation Map (CAM) methods to the medical imaging diagnosis discussed in the example. After the first published paper on Class Activation Map, advances and improvements were made: Gradient-weighted Class Activation Mapping (Grad-CAM) which relaxed the constraints of the network architecture with the help of the gradient of the output with respect to the activations, Grad-CAM++ which achieves improvements in multiple locations of the same class and Smooth Grad-CAM++ which incorporates more images identical to the original with added noise to improve the decision making and smooth the activation map. These techniques have been implemented and can be accessed via <https://github.com/pedrogallegolpz/TFG>.

We have, therefore, four explainability techniques: each one is supposed to be more powerful than its predecessor. In this study we are going to study which technique behaves better and is more manageable in this problem, making an depth analysis of how they behave. These techniques are applied on existing models, so in order not to bias the results, they will be studied by taking four different base convolutional networks: VGG16, RESNET18, MOBILENETv2 and EFFICIENTNETbo. This results in four results for each technique and four results for each model. Although the CAM method is shown in the study, it will not be evaluated in the comparison between techniques since it has an architecture restriction (it can only have at most one fully connected layer in its classifier), Grad-CAM being a direct generalisation of CAM for models with architectures that can have more than one fully connected layer in the classifier.

The study is performed on a publicly accessible prostate histological image dataset: SICAPv1. These are large images of 79 patients. The source of the dataset ensures pixel accuracy for the labelling of the masks. However, it is possible to get examples where the mask picks up areas where there is background as tissue. The images have had to be processed to decrease their size and consequently cause an increase in the number of instances of the dataset to be used. This image processing causes problems in the study of explainability, creating bias in the dataset with images that have a large percentage of pathological tissue in their domain and causing inconsistencies in the location of pathological tissue in contiguous images. These problems coupled with

inaccurate labelling of the masks will lead to an error in the localisation metrics of the study.

Problems such as those discussed with the dataset, coupled with the intrinsic localisation limitations of these techniques, will cause the quality of the results to be strictly limited to less than perfect. The intrinsic limitations of the techniques have to do with the resizing of the activation map or heat map: by resizing to a larger size (the dimensions of the input), pixel precision is lost, making it impossible to adjust to any region where there is pathological tissue.

The proposal made in the work is to make a separate comparison between models and techniques. The idea is to take the best network among the four by making an overall assessment of the techniques on each model. On the other hand, taking the best technique among the four, a global evaluation of the models will be made on each technique. At the end there will be one best model and one best technique, which will be the winning pair. This implies that the best pair may be the pair that does not have the best mark in any of the metrics on which they are evaluated. In terms of results, one would expect the best model to be the one with the most innovative architecture and the best technique to be the one that has incorporated the most improvements. But no, the classical model based on VGG16 is the best together with the Grad-CAM base technique. In this decision, computational times were important, as there was no clear winner among the techniques.

It can be seen that factors such as the classification capability of a model directly influence the localisation capability of these techniques. The Smooth Grad-CAM++ technique offers a strong dependence on the model and the dataset, which makes it unsuitable both for its computational time and the difficulty of parameterising it well.

Finally, the solutions provided by the best combination of model and technique are of sufficient quality to be incorporated into the day-to-day work of a doctor and thus to value the help that this tool offers to the expert. The avenues for future work are multiple and this is just a small approximation of what Artificial Intelligence can do to help in an essential area such as medicine.

A test code of the tool is available within Google Colab, which can be accessed from the project's GitHub: <https://github.com/pedrogallegolpz/TFG>. There you can run more examples than the ones shown in this work (appendix A).

ÍNDICE GENERAL

INTRODUCCIÓN	12
I. FUNDAMENTOS MATEMÁTICOS	16
1. FUNDAMENTOS MATEMÁTICOS	17
1.1. Álgebra Lineal	18
1.1.1. Operaciones con matrices	19
1.1.2. Espacios vectoriales	22
1.1.3. Aplicaciones lineales	27
1.2. Diferenciabilidad	28
1.2.1. Definición de función diferenciable	28
1.2.2. Propiedades de la diferencial	29
1.2.3. Regla de la cadena	30
1.2.4. Vector Gradiente	32
1.3. Optimización	34
1.3.1. Descenso de Gradiente	34
1.4. Probabilidad	34
1.4.1. Fundamentos de la probabilidad	35
1.4.2. Desigualdad de Hoeffding	37
II. TEORÍA INFORMÁTICA	41
2. INTELIGENCIA ARTIFICIAL	42
2.1. Introducción a la Inteligencia Artificial	42
2.2. Aprendizaje Automático	43
2.2.1. Fundamentos	44
2.2.2. Perceptrón	51
2.2.3. Perceptrón Multicapa - Red Neuronal	52
2.2.4. Deep Learning	56
2.3. Redes Convolucionales	57
2.3.1. Arquitectura de una Red Convolutacional	58
2.3.2. Clasificación en Redes Convolucionales	60
2.3.3. Detección de objetos	60
3. XAI EXPLAINABLE AI	62
3.1. XAI	62
3.2. Class Activation Map	63
3.2.1. Idea	64
3.2.2. Método	64
3.3. Gradient-weighted Class Activation Mapping	66

3.3.1. Método	66
3.3.2. Generalización de CAM	66
3.4. Grad-CAM++	68
3.4.1. Método	68
3.4.2. Análisis computacional	69
3.5. Smooth Grad-CAM++	70
3.5.1. Smooth Grad	71
3.5.2. Método	72
III. APLICACIÓN	73
4. APLICACIÓN DE LAS TÉCNICAS DE EXPLICABILIDAD AL DIAGNÓSTICO DE CÁNCER EN IMÁGENES HISTOLÓGICAS DE PRÓSTATA	74
4.1. Dataset	74
4.1.1. Procesado del dataset	75
4.2. Redes utilizadas	79
4.2.1. Implementación	80
4.3. Entrenamiento y optimización de parámetros	81
4.3.1. Entrenamiento	82
4.3.2. Optimización de parámetros de Smooth Grad-CAM++	88
4.3.3. Optimización de umbral de máscara	89
4.4. Análisis de métricas sobre Test	91
4.4.1. Métricas de clasificación	91
4.4.2. Métricas de localización	92
4.4.3. Comparación de modelos y resultados	95
4.5. Conclusión	108
4.6. Trabajo futuro	108
A. APÉNDICE A. EJEMPLOS SOBRE EL CONJUNTO DE TEST	113

INTRODUCCIÓN

La explosión de la Inteligencia Artificial en el mundo llegó hace una década, cuando profesiones de cualquier disciplina empezaron a ver su potencial e intentaron buscarle aplicación en su sector. Esta explosión llegó de la mano de los modelos matemáticos de Aprendizaje Automático. Fue de hecho el concepto *aprendizaje de las máquinas* lo que provocó una revolución, rompiendo una barrera de ficción creada por los humanos muchos años atrás. Se vio cómo los modelos de Aprendizaje Profundo eran capaces de resolver tareas muy complejas y con un gran desempeño. Un ejemplo podría ser el de un modelo de Red Convolutacional resolviendo un problema de diagnóstico de imagen médica, donde un humano no experto en la materia sería incapaz de resolverlo. Esta potencia de resultados hizo que la confianza puesta en esta tecnología aumentase. Ahora, siguiendo con el ejemplo, los médicos podrían usar esa herramienta para diagnosticar, ahorrándoles esfuerzo y tiempo.

Sin embargo, la Inteligencia Artificial chocaría con una barrera: la explicabilidad [1]. Determinar qué decisión tomar cuando el médico tenía una opinión en el diagnóstico distinta a la opinión del modelo de aprendizaje profundo se volvió todo un problema. La complejidad de los modelos hacía imposible entender por qué tomaban esas decisiones, considerándolos cajas negras, completamente opacas. Esto provocaba que el médico tomara las decisiones únicamente bajo su criterio, ya que si tanto el médico como el modelo daban el mismo diagnóstico entonces se seguía su propio criterio; pero si el médico daba un diagnóstico distinto al del modelo, era muy probable que el médico siguiera su propio diagnóstico porque lo entendía, dando por sentado que el modelo podía haberse equivocado en ese caso.

Así, surgió la llamada XAI[1], Inteligencia Artificial eXplicable, cuya misión era proporcionar herramientas de Inteligencia Artificial asegurando transparencia en estos modelos. Este nuevo campo de estudio abría una gran variedad puertas, como la ética dentro de la Inteligencia Artificial, pudiendo valorarse ahora la ética dentro del campo de decisiones del modelo. Otras puertas como la explicabilidad para expertos en el área de aplicación del modelo (como el médico), permitiendo entender los resultados para poder ayudar a su propio criterio. La explicabilidad también era útil para expertos en Inteligencia Artificial (científicos de datos por ejemplo), detectando fallos del modelo como sesgos u otro tipo de errores que pueden afectar a la bondad de los resultados.

Los métodos de Class Activation Map [2] son herramientas surgidas de XAI[1] en el área de Visión por Computador. Estos métodos o técnicas permiten entender en qué se ha basado el modelo para dar los resultados que ha arrojado. Visualmente

crean un mapa de activación sobre la entrada, resaltando aquellas regiones que han sido especialmente relevantes para la toma de la decisión. En el problema del diagnóstico en imágenes médicas, estas técnicas conseguirían determinar qué partes de la imagen han sido las más relevantes para el diagnóstico. Esto significa que si el diagnóstico fuese positivo (hay presencia de patología), las técnicas serían capaces de determinar la región donde se encontrasen esas características que marcaran la presencia de patología. Así, estas técnicas tienen mucha potencia ya que con ellas se consiguiría ver que los modelos entrenados para la clasificación son capaces de localizar de manera no supervisada. Ahora, en este tipo de diagnóstico basado en imagen, un médico podría usar estos modelos con criterio, usándolo de complemento a sus conocimientos.

El objeto de estudio de este trabajo es precisamente la aplicación de los métodos de Class Activation Map (CAM) al diagnóstico de imágenes médicas del que se ha hablado en el ejemplo. Tras el primer artículo publicado de Class Activation Map[2], surgieron avances y mejoras: Gradient-weighted Class Activation Mapping (Grad-CAM) [3] que relajaba las restricciones de la arquitectura de la red con ayuda del gradiente de la salida con respecto a las activaciones, Grad-CAM++ [4] que logra mejoras en localizaciones múltiples de una misma clase y Smooth Grad-CAM++[5] que incorpora más imágenes idénticas a la original con ruido añadido para robustecer la toma de decisión. Estas técnicas han sido implementadas desde cero y se puede tener acceso a su implementación a través de <https://github.com/pedrogallegolpz/TFG>.

Los modelos matemáticos de Aprendizaje Profundo a los que se aplican estas técnicas, así como las técnicas en sí tienen una gran base matemática: se hará uso de conceptos del álgebra lineal [6] así como operaciones matriciales para poder computar los valores que arrojan los modelos. Será necesario el análisis matemático [7] para hacer consistente el uso de Gradienes y poder utilizarlos en problemas de optimización, elemento fundamental para el aprendizaje de los modelos. Además se necesitarán unas bases de probabilidad [8] que darán la certeza de que los modelos pueden aprender de los datos con la desigualdad de Hoeffding. Por otro lado, estos conceptos también serán usados por las técnicas de explicabilidad, las cuales hacen un uso importante (menos CAM[2]) de los gradienes del modelo. Una vez sentadas las bases matemáticas del aprendizaje se pasará a estudiar conceptos clave de esta rama de la informática, la Inteligencia Artificial [9]: error, funciones de pérdida, criterio de aprendizaje (método de optimización), compromiso sesgo-varianza, regularización, tipos de problemas... De ahí se pasará a estudiar los modelos de Perceptrón[9] y Perceptrón multicapa[9] que darán pie al Aprendizaje Profundo [10]. Las Redes Convolucionales serán el eje de giro del trabajo a partir de este punto, ya que será el tipo de modelo utilizado para la aplicación final del trabajo.

Con todas las bases asentadas se pasará a la aplicación de toda la teoría estudiada. Se tienen, por lo tanto, cuatro técnicas de explicabilidad: donde cada una se supone es

más potente que su antecesora. En este trabajo se va a estudiar qué técnica se comporta mejor y es más manejable en este problema, haciendo un análisis profundo de las métricas de localización y clasificación. Estas técnicas se aplicarán sobre cuatro redes distintas: VGG16 [11], RESNET18[12], MOBILENETv2[13] y EFFICIENTNETbo[14]. De esta manera se conseguirá ver el desempeño de las técnicas (que deben ser independiente del modelo) de manera global entre todos los modelos, eliminando así de la evaluación de las técnicas el posible efecto que pueda tener el modelo concreto usado en el problema. Esto da lugar a cuatro resultados por cada técnica y a cuatro resultados por cada modelo. Aunque el método de CAM[2] se muestre en el estudio, este no va a valorarse en la comparación entre técnicas puesto que posee una restricción de arquitectura (solo puede tener como máximo una capa totalmente conectada en su clasificador), siendo Grad-CAM[3] una generalización directa de CAM[2] para modelos con arquitecturas que pueden tener más de una capa totalmente conectada en el clasificador.

El estudio se realiza sobre un conjunto de datos de imágenes histológicas de próstata a la que se tiene un acceso público: SICAPv1. Son imágenes de gran tamaño de 79 pacientes. En la fuente del conjunto de datos se asegura una precisión de píxel para el etiquetado de las máscaras. Sin embargo, existen imágenes del conjunto de datos donde la máscara etiqueta como tejido donde hay fondo. Las imágenes han tenido que ser procesadas para disminuir su tamaño y, consecuentemente, provocar un aumento en el número de instancias del conjunto de datos que se va a usar. Este procesado de imágenes da problemas en el estudio de explicabilidad, sesgando el conjunto de datos con imágenes que poseen un gran porcentaje de tejido patológico en su dominio y provocando incoherencias de localización de tejido patológico en imágenes contiguas. Estos problemas unidos a la imprecisión del etiquetado de las máscaras acarrearán un error en las métricas de localización del estudio.

Problemas como los comentados con el conjunto de datos, unidos a las limitaciones de localización que poseen estas técnicas de manera intrínseca, van a provocar que la calidad de los resultados esté acotada estrictamente por debajo de la perfección. Las limitaciones intrínsecas de las técnicas tienen que ver con la redimensionación que hacen del mapa de activación o mapa de calor: al redimensionar a un tamaño mayor (las dimensiones de la entrada) se pierde precisión de píxel, haciendo imposible ajustarse a todos los casos de región donde haya tejido patológico.

La propuesta del trabajo es hacer una comparación por separado entre modelos y técnicas. La idea es coger la mejor red entre las cuatro haciendo una valoración global de las técnicas sobre cada modelo. Por otro lado, cogiendo la mejor técnica entre las cuatro se va a hacer una valoración global de los modelos sobre cada técnica. Al final se tendrá un mejor modelo y una mejor técnica, la cual será la pareja ganadora. Este proceso de decisión, implica que se puede dar que la mejor pareja no sea la pareja que tenga la mejor marca en alguna de las métricas sobre las que se evalúan. En cuanto a los resultados cualquiera esperaría que el mejor modelo fuese aquel que posee una

arquitectura más novedosa y que la mejor técnica fuese aquella que ha incorporado más mejoras. Pero no, al contrario, el modelo clásico basado en VGG16[11] es el mejor junto con la técnica base de Grad-CAM[3]. En esta decisión los tiempos de cómputo tuvieron importancia, ya que, dentro de las técnicas no había un claro ganador.

Se consigue ver como algunos factores como la capacidad de clasificación de un modelo influye de manera directa sobre la capacidad de localización a través de estas técnicas. La técnica de Smooth Grad-CAM++[5] ofrece una dependencia fuerte del modelo y del conjunto de datos, lo que hace que no sea una técnica apropiada tanto por su tiempo de cómputo como por la dificultad de parametrizarla bien.

Por último, las soluciones dadas por la mejor pareja de modelo y técnica tienen una calidad suficiente para poder incorporarse al día a día de un médico y así valorar la ayuda que esta herramienta ofrece al experto. Las vías de trabajo futuro son múltiples y esto es solo una pequeña aproximación de lo que se puede ayudar con la Inteligencia Artificial a un área esencial como lo es la medicina.

Se dispone de un código de prueba de la herramienta dentro de Google Colab, al se puede acceder desde el GitHub del proyecto: <https://github.com/pedrogallegolpz/TFG>. Allí se podrán ejecutar más ejemplos que los mostrados en este trabajo (apéndice A)

Parte I

FUNDAMENTOS MATEMÁTICOS

Bases teóricas matemáticas del trabajo. Se hablará de Álgebra lineal para las operaciones matriciales, los espacios normados y las aplicaciones lineales. Se tocará la diferenciabilidad y la optimización. Por último, la teoría de la probabilidad dará la certeza con la desigualdad de Hoeffding, de que el concepto de aprendizaje desarrollado en la teoría de Inteligencia Artificial tiene sentido.

FUNDAMENTOS MATEMÁTICOS

En este capítulo se introducirá al lector a las bases matemáticas que sustentan toda la teoría de las técnicas de explicabilidad en imágenes, objeto final de este trabajo.

René Descartes, filósofo, matemático y físico francés considerado el padre de la geometría analítica y la filosofía moderna [15], así como uno de los protagonistas con luz propia en el umbral de la revolución científica concibe las matemáticas como:

“La ciencia del orden y la medida, de bellas cadenas de razonamientos, todos sencillos y fáciles”.

Murray Gell-Mann, físico estadounidense que recibió el Premio Nobel de Física en 1969 por sus descubrimientos sobre partículas elementales [16], define las matemáticas como:

“El estudio riguroso de mundos hipotéticos. Es la ciencia de lo que podría haber sido o podría ser, así como de lo que es”.

La informática siempre ha estado estrechamente ligada a las matemáticas. Visión por Computador es una de las áreas de la informática que hace uso de herramientas matemáticas, siendo muy común el uso de **Redes Convolucionales** donde su estructura, su capacidad funcional y sus resultados parten de unas herramientas matemáticas. Estas redes serán el concepto informático base utilizado en este trabajo. Las operaciones matriciales pueden verse como la herramienta básica de estas redes y es aquí donde interviene el **Álgebra Lineal**. Pero no será la única intervención de esta rama, ya que con ella se pueden introducir el concepto de norma y aplicación lineal. Éstos, son conceptos esenciales que además de introducir otras herramientas matemáticas, son conceptos usados también en inteligencia artificial (entre otras cosas para regularizar modelos).

El proceso de aprendizaje de un modelo se puede medir con el número de errores del mismo. A modo de ejemplo ilustrativo, se puede imaginar cómo se evalúan los conocimientos de los alumnos tradicionalmente. En un examen, un alumno tiene asociada una nota, la cual será menor cuantos más errores tenga y mayor cuantos

más aciertos. Por lo tanto, el objetivo del alumno es *minimizar* su error en el examen. Aquí se introduce el concepto de **Optimización**, que nos dará de herramienta de minimización el algoritmo de **Descenso de Gradiente**. Es un algoritmo que hará uso de otra herramienta clave: la **Diferenciabilidad**, donde el concepto de *gradiente* cobra especial relevancia. Los gradientes son la aplicación de la diferenciabilidad a campos escalares, concretamente serán usados sobre espacios normados en las redes convolucionales con dos objetivos principales:

1. Optimizar, con el algoritmo de Descenso por Gradiente.
2. Explicar los resultados arrojados por el modelo.

Por otro lado, la **Regla de la Cadena**, resultado en el estudio de la Diferenciabilidad, permitirá definir el algoritmo de **backpropagation**. Este algoritmo se utiliza para propagar de forma eficiente el gradiente del error desde la salida del modelo hasta la entrada. Así el efecto del algoritmo de optimización de Descenso por Gradiente se propaga hasta la entrada, permitiendo a las redes profundas aprender.

Como última sección se introducen fundamentos de **Probabilidad**, herramienta usada para tratar la incertidumbre en la disciplina del Aprendizaje Automático. Destaca la **Desigualdad de Hoeffding** dentro de esta sección por su relevancia que tendrá en la parte informática. Esta desigualdad aplicada al Aprendizaje permitirá establecer una cota de error cuyo significado es muy importante: se puede aprender a través de los datos.

1.1 ÁLGEBRA LINEAL

El **Álgebra Lineal** es una rama de las matemáticas ampliamente usada en las ingenierías y disciplinas técnicas. En concreto, es usada en Aprendizaje Automático, lo que hace esencial comprender esta sección porque dará herramientas que serán utilizadas más adelante en este trabajo. Con el apoyo de [6] y [10] se introducirán las bases del álgebra lineal, estudiando desde las operaciones con matrices hasta las aplicaciones lineales, pasando por los espacios normados.

En primer lugar, se introducen los elementos básicos con los que se va a trabajar. Hablaremos de **matriz** de orden $m \times n$ con coeficientes en \mathbb{R} cuando se hablen de elementos de $\mathbb{R}^{m,n}$. Un elemento $\mathbf{A} \in \mathbb{R}^{m,n}$ se representa como:¹

¹ \mathbb{R}^n hace referencia al producto cartesiano de \mathbb{R} n veces $\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times \cdots (n) \cdots \times \mathbb{R}$. Por otro lado $\mathbb{R}^{m,n}$ hace referencia al producto cartesiano de $\mathbb{R}^m \times \mathbb{R}^n$, que sería equivalente a $\mathbb{R}^{m \times n}$, pero se nota como $\mathbb{R}^{m,n}$ cuando se quiere resaltar que se trabaja con dos dimensiones principales. De la misma manera se podría hablar de $\mathbb{R}^{n_1, n_2, \dots, n_k}$, $k \in \mathbb{N}$ para resaltar k dimensiones principales.

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}$$

Si $m = 1$ se llamará *matriz fila*. Si $n = 1$ se llamará *matriz columna*. Si $m = n$ se llamará *matriz cuadrada*. Dos matrices \mathbf{A}, \mathbf{B} son iguales cuando tienen el mismo orden y sus elementos coinciden entre ellos en todas las posiciones.

1.1.1 Operaciones con matrices

En esta sección se definen las operaciones necesarias para dotar al conjunto de las matrices de estructura algebraica.

1.1.1.1 Suma de Matrices

Dadas dos matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m,n}$, se define su suma como:

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} A_{1,1} + B_{1,1} & A_{1,2} + B_{1,2} & \cdots & A_{1,n} + B_{1,n} \\ A_{2,1} + B_{2,1} & A_{2,2} + B_{2,2} & \cdots & A_{2,n} + B_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} + B_{m,1} & A_{m,2} + B_{m,2} & \cdots & A_{m,n} + B_{m,n} \end{bmatrix}$$

esto es, la suma de las matrices es la suma por posición de sus elementos. Nótese que la suma de matrices está definida solo para matrices de igual orden. Esta operación verifica las propiedades de:

- **Asociativa:**

$$\begin{aligned} \mathbf{A} + (\mathbf{B} + \mathbf{C}) &= \mathbf{A} + [B_{i,j} + C_{i,j}] \\ &= [A_{i,j} + (B_{i,j} + C_{i,j})] \\ &= [(A_{i,j} + B_{i,j}) + C_{i,j}] \\ &= [(A_{i,j} + B_{i,j})] + \mathbf{C} \\ &= (\mathbf{A} + \mathbf{B}) + \mathbf{C}, \quad \forall \mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{m,n} \end{aligned}$$

- **Commutativa:**

$$\mathbf{A} + \mathbf{B} = [A_{i,j} + B_{i,j}] = [B_{i,j} + A_{i,j}] = \mathbf{B} + \mathbf{A}, \quad \forall \mathbf{A}, \mathbf{B} \in \mathbb{R}^{m,n}$$

- **Elemento neutro:** Si se define la matriz $0^{m,n}$ como aquella que es 0 en todos sus elementos: $0_{i,j}^{m,n} = 0 \in \mathbb{R}$, se tiene que es el elemento neutro:

$$\mathbf{A} + 0^{m,n} = [0_{i,j}^{m,n} + A_{i,j}] = [0 + A_{i,j}] = [A_{i,j}] = \mathbf{A}, \quad \forall \mathbf{A} \in \mathbb{R}^{m,n}$$

- **Elementos simétricos:** Dada la matriz \mathbf{A} , si se define la matriz $B = -A_{i,j} \in \mathbb{R}$, se tiene que:

$$\forall \mathbf{A} \in \mathbb{R}^{m,n}, \quad \exists \mathbf{B} \in \mathbb{R}^{m,n} \text{ tal que } \mathbf{A} + \mathbf{B} = [B_{i,j} + A_{i,j}] = [-A_{i,j} + A_{i,j}] = [0] = 0^{m,n}$$

1.1.1.2 Producto de un escalar por una matriz

Dada una matriz $\mathbf{A} \in \mathbb{R}^{m,n}$ y dado un escalar $a \in \mathbb{R}$, se define el producto como:

$$a\mathbf{A} = \mathbf{A}a = \begin{bmatrix} aA_{1,1} & aA_{1,2} & \cdots & aA_{1,n} \\ aA_{2,1} & aA_{2,2} & \cdots & aA_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ aA_{m,1} & aA_{m,2} & \cdots & aA_{m,n} \end{bmatrix}$$

El producto de un escalar por una matriz verifica las siguientes propiedades:

- **Distributiva respecto de la suma de escalares:**

$$(a+b)\mathbf{A} = [(a+b)A_{i,j}] = [aA_{i,j} + bA_{i,j}] = a\mathbf{A} + b\mathbf{A}, \quad \forall \mathbf{A} \in \mathbb{R}^{m,n}; \quad \forall a, b \in \mathbb{R}$$

- **Distributiva respecto de la suma de matrices:**

$$a(\mathbf{A} + \mathbf{B}) = [a(A_{i,j} + B_{i,j})] = [aA_{i,j} + aB_{i,j}] = a\mathbf{A} + a\mathbf{B}, \quad \forall \mathbf{A}, \mathbf{B} \in \mathbb{R}^{m,n}; \quad \forall a \in \mathbb{R}$$

- **Pseudoasociativa:**

$$(ab)\mathbf{A} = [(ab)A_{i,j}] = [a(bA_{i,j})] = a[bA_{i,j}] = a(b\mathbf{A}), \quad \forall \mathbf{A} \in \mathbb{R}^{m,n}; \quad \forall a, b \in \mathbb{R}$$

- **Ley de identidad:**

$$1^{m,n}\mathbf{A} = [1A_{i,j}] = [A_{i,j}] = \mathbf{A}, \quad \forall \mathbf{A} \in \mathbb{R}^{m,n}$$

1.1.1.3 Producto de matrices

Dadas matrices $\mathbf{A} \in \mathbb{R}^{m,p}$, $\mathbf{B} \in \mathbb{R}^{p,n}$, se define su producto como

$$\mathbf{AB} = [C_{i,j}] = \mathbf{C} \in \mathbb{R}^{m,n}$$

donde:

$$C_{i,j} = \sum_{k=1}^p A_{i,k}B_{k,j}$$

Así pues, el producto de matrices está solamente definido cuando el número de columnas de la primera matriz \mathbf{A} es el mismo que el número de filas de la segunda matriz \mathbf{B} . Para las propiedades del producto de matrices, se van a usar las matrices $\mathbf{A}, \mathbf{A}' \in \mathbb{R}^{m,p}; \quad \mathbf{B}, \mathbf{B}' \in \mathbb{R}^{p,n}; \quad \mathbf{C} \in \mathbb{R}^{n,q}$ y el escalar $a \in \mathbb{R}$.

1. $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$.

$$\begin{aligned}\mathbf{A}(\mathbf{BC}) &= \mathbf{A} \left(\left[\sum_l B_{i,l} C_{l,j} \right] \right) = \left[\sum_k \left(A_{i,k} \sum_l B_{k,l} C_{l,j} \right) \right] \\ &= \left[\sum_k \sum_l A_{i,k} B_{k,l} C_{l,j} \right] = \left[\sum_l \sum_k A_{i,k} B_{k,l} C_{l,j} \right] \\ &= \left[\sum_l \left(\sum_k A_{i,k} B_{k,l} \right) C_{l,j} \right] = \left[\left(\sum_k A_{i,k} B_{k,l} \right) \right] \mathbf{C} = (\mathbf{AB})\mathbf{C}\end{aligned}$$

2. $\mathbf{I}^m \mathbf{A} = [\sum_{l=0}^m \delta_{i,l} A_{l,j}] = [A_{i,j}] = \mathbf{A} = [A_{i,j}] = [\sum_{l=0}^p A_{i,l} \delta_{l,j}] = \mathbf{A} \mathbf{I}^p$ ².

3. $\mathbf{A}(\mathbf{B} + \mathbf{B}') = \left[\sum_k A_{i,k} (B_{k,j} + B'_{k,j}) \right] = [\sum_k A_{i,k} B_{k,j} + \sum_k A_{i,k} B'_{k,j}] = \mathbf{AB} + \mathbf{AB}'$.

4. $(\mathbf{A} + \mathbf{A}')\mathbf{B} = \left[\sum_k (A_{i,k} + A'_{i,k}) B_{k,j} \right] = [\sum_k A_{i,k} B_{k,j} + \sum_k A'_{i,k} B_{k,j}] = \mathbf{AB} + \mathbf{A}'\mathbf{B}$.

5. $a(\mathbf{AB}) = [a \sum_k A_{i,k} B_{k,j}] = [\sum_k (a A_{i,k}) B_{k,j}] = (a\mathbf{A})\mathbf{B}$.

1.1.1.4 Matriz traspuesta

Una operación importante de las matrices es la trasposición. Dada una matriz $\mathbf{A} \in \mathbb{R}^{m,n}$ se llamará **matriz traspuesta** de \mathbf{A} a la matriz \mathbf{A}^T que cumple que

$$\mathbf{A}_{i,j} = \mathbf{A}_{j,i}^T$$

Donde $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$. La matriz \mathbf{A} se dirá que es **simétrica** si $\mathbf{A} = \mathbf{A}^T$. Dadas matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m,p}; \quad \mathbf{C} \in \mathbb{R}^{p,n}$; y el escalar $a \in \mathbb{R}$, las propiedades son:

1. $(\mathbf{A} + \mathbf{B})^T = [A_{i,j} + B_{i,j}]^T = [A_{j,i} + B_{j,i}] = [A_{i,j}^T + B_{i,j}^T] = \mathbf{A}^T + \mathbf{B}^T$.

2. $(\mathbf{AC})^T = [\sum_k^p A_{i,k} C_{k,j}]^T = [\sum_k^p A_{j,k} C_{k,i}] = [\sum_k^p C_{i,k}^T A_{k,j}^T] = \mathbf{C}^T \mathbf{A}^T$.

3. $(a\mathbf{A})^T = [a A_{i,j}]^T = a[A_{i,j}]^T = a\mathbf{A}^T$.

² A la matriz identidad de tamaño n se denota como $\mathbf{I}^n = [\delta_{i,j}]$, donde se ha usado la δ de Kronecker.

$$\delta_{i,j} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

1.1.1.5 Matriz inversa

Dadas matrices cuadradas $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n,n}$, se dice que \mathbf{B} es la **matriz inversa** de \mathbf{A} si $\mathbf{AB} = \mathbf{BA} = \mathbf{I}^{n,n}$. Por lo tanto, diremos que $\mathbf{A} \in \mathbb{R}^{n,n}$ es **invertible** si \mathbf{A} tiene inversa.

Lema 1. Una matriz invertible $\mathbf{A} \in \mathbb{R}^{n,n}$ tiene una única inversa

Demostración. Si \mathbf{A} tuviese dos inversas \mathbf{B} y \mathbf{C} , entonces:

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}^{n,n} = \mathbf{CA} = \mathbf{AC}$$

De donde:

$$\mathbf{B} = \mathbf{BI}^{n,n} = \mathbf{B}(\mathbf{AC}) = (\mathbf{BA})\mathbf{C} = \mathbf{I}^{n,n}\mathbf{C} = \mathbf{C}$$

□

Propiedades de la matriz inversa.

1. Si \mathbf{A} y \mathbf{B} son invertibles, entonces \mathbf{AB} es invertible y $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$

$$(\mathbf{AB})(\mathbf{B}^{-1}\mathbf{A}^{-1}) = \mathbf{A}(\mathbf{BB}^{-1})\mathbf{A}^{-1} = \mathbf{A}\mathbf{I}^{n,n}\mathbf{A}^{-1} = \mathbf{I}^{n,n}$$

$$(\mathbf{B}^{-1}\mathbf{A}^{-1})(\mathbf{AB}) = \mathbf{B}^{-1}(\mathbf{A}^{-1}\mathbf{A})\mathbf{B} = \mathbf{B}^{-1}\mathbf{I}^{n,n}\mathbf{B} = \mathbf{I}^{n,n}$$

Con lo que \mathbf{AB} es invertible y su inversa es $\mathbf{B}^{-1}\mathbf{A}^{-1}$. Análogamente, se puede deducir para k matrices: $\mathbf{A}_1, \dots, \mathbf{A}_k \in \mathbb{R}^{n,n}$, se tiene que $\mathbf{A}_1 \cdots \mathbf{A}_k$ es invertible y $(\mathbf{A}_1 \cdots \mathbf{A}_k)^{-1} = \mathbf{A}_k^{-1} \cdots \mathbf{A}_1^{-1}$.

2. Si \mathbf{A} es invertible, entonces \mathbf{A}^T es invertible y $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$.

$$\mathbf{A}^T(\mathbf{A}^{-1})^T = (\mathbf{A}^{-1}\mathbf{A})^T = (\mathbf{I}^{n,n})^T = \mathbf{I}^{n,n}$$

$$(\mathbf{A}^{-1})^T\mathbf{A}^T = (\mathbf{AA}^{-1})^T = (\mathbf{I}^{n,n})^T = \mathbf{I}^{n,n}$$

Con lo que \mathbf{A}^T es invertible y su inversa es $(\mathbf{A}^{-1})^T$.

1.1.2 Espacios vectoriales

Se introduce ahora la definición de *espacio vectorial* que dará pie a conceptos relevantes como los *espacios normados* o las *aplicaciones lineales*.

Definición 1. Siendo V un conjunto no vacío, se dice que V es un **espacio vectorial** sobre \mathbb{R} si:

1. En V hay definida una operación interna (denotada por $+$, de forma que $(V, +)$ verifica las propiedades:

- a) *Asociativa:* $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w}), \quad \forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V$
- b) *Commutativa:* $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}, \quad \forall \mathbf{u}, \mathbf{v} \in V$
- c) *Existencia de elemento neutro:* $\exists \mathbf{0} \in V$ tal que $\mathbf{0} + \mathbf{v} = \mathbf{v}, \quad \forall \mathbf{v} \in V$
- d) *Existencia de elemento opuesto:* $\forall \mathbf{v} \in V, \quad \exists (-\mathbf{v}) \in V$ tal que $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
2. En V hay definida una operación externa de \mathbb{R} en V , que se denotará como yuxtaposición, verificando:
- a) $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}, \quad \forall a \in \mathbb{R}, \quad \forall \mathbf{u}, \mathbf{v} \in V$
- b) $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}, \quad \forall a, b \in \mathbb{R}, \quad \forall \mathbf{u} \in V$
- c) $a(b\mathbf{u}) = (ab)\mathbf{u}, \quad \forall a, b \in \mathbb{R}, \quad \forall \mathbf{u} \in V$
- d) $1\mathbf{u} = \mathbf{u}, \quad \forall \mathbf{u} \in V$

Los elementos del espacio vectorial suelen denominarse **vectores**, mientras que los elementos de \mathbb{R} se les conoce como **escalares**.

1.1.2.1 Bases de un espacio vectorial

Dado un conjunto de vectores $B = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ en un \mathbb{R} -espacio vectorial V , se dice que u es una *combinación lineal* de estos vectores si $\mathbf{u} = a_1\mathbf{v}_1 + \dots + a_n\mathbf{v}_n$ con a_i escalares en $\mathbb{R}, \quad \forall i \in \{1, \dots, n\}$. Entonces, B es:

- Un conjunto *linealmente dependiente* si el vector $\mathbf{0}$ se puede expresar como combinación lineal de ellos. En caso contrario se dirá que son *linealmente independientes*.
- Un *sistema de generadores* de V si todo vector de V es combinación lineal de ellos.

Si B es *linealmente independiente* y además es un *sistema de generadores*, se dirá que B es una **base** de V .

Teorema 1. Si un espacio vectorial V tiene una base formada por un número finito de vectores, entonces todas las bases de V son finitas y tienen el mismo número de vectores.

Demostración. Sea $B = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ base de V , y sea B' otra base de V . Por ser B' linealmente independiente ha de tener como máximo n vectores. Ahora, puesto que B' es sistema de generadores y en B hay n vectores linealmente independientes, B' ha de tener al menos n vectores. Concluyendo que B' tiene n vectores. \square

De un espacio vectorial V que posee una base finita se dirá que es un **espacio vectorial de dimensión finita** y llamando **dimensión** de V ($\dim V$) al número de vectores de cualquier base.

De aquí en adelante se notará como **base usual** a la base $B = \{e_1, e_2, \dots, e_n\}$ donde $e_k = \{0_1, 0_2, \dots, 0_{k-1}, 1_k, 0_{k+1}, \dots, 0_n\}$

1.1.2.2 Coordenadas de un vector respecto de una base

Proposición 1. *Sea V un espacio vectorial sobre \mathbb{R} . Si $B = \{\mathbf{u}_1, \dots, \mathbf{u}_n\}$ es una base de V , entonces todo vector $\mathbf{x} \in V$ se escribe de forma única como combinación lineal de los vectores de B .*

Demostración. Dado que B es un sistema de generadores, todo vector de V se escribe como combinación lineal de los vectores de B , falta ver la unicidad. Si se ve que no es única la combinación lineal y se llega a contradicción se tendría demostrado. Si la combinación no es única, entonces:

$$\begin{aligned}\mathbf{x} &= x_1 \mathbf{u}_1 + x_2 \mathbf{u}_2 + \cdots + x_n \mathbf{u}_n \\ \mathbf{x} &= x'_1 \mathbf{u}_1 + x'_2 \mathbf{u}_2 + \cdots + x'_n \mathbf{u}_n\end{aligned}$$

Entonces,

$$\begin{aligned}x_1 \mathbf{u}_1 + x_2 \mathbf{u}_2 + \cdots + x_n \mathbf{u}_n &= x'_1 \mathbf{u}_1 + x'_2 \mathbf{u}_2 + \cdots + x'_n \mathbf{u}_n \\ &\Downarrow \\ (x_1 - x'_1) \mathbf{u}_1 + (x_2 - x'_2) \mathbf{u}_2 + \cdots + (x_n - x'_n) \mathbf{u}_n &= 0\end{aligned}$$

Como B es linealmente independiente, el vector 0 no se puede escribir como combinación lineal de los elementos de B , así que $x_i - x'_i = 0 \iff x_i = x'_i, \quad \forall i \in \{1, \dots, n\}$, luego las dos expresiones eran idénticas. \square

Después de este resultado, se tiene que dado un vector y una base, el vector puede representarse por los escalares que aparecen en la anterior combinación lineal. Esta herramienta permitirá trabajar en cualquier espacio vectorial de dimensión n como en el espacio \mathbb{R}^n . Sea $B = \{\mathbf{u}_1, \dots, \mathbf{u}_n\}$ base de V , si $\mathbf{x} = x_1 \mathbf{u}_1 + x_2 \mathbf{u}_2 + \cdots + x_n \mathbf{u}_n$ es la expresión única del vector $\mathbf{x} \in V$, como combinación lineal de vectores de la base B , se dice entonces que (x_1, \dots, x_n) son las **coordenadas** de \mathbf{x} respecto de la base B . Y se representará como:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

Tratando a los vectores como *matrices fila* o *matrices columna*, se pueden inferir de las matrices las operaciones que se pueden hacer con los vectores.

1.1.2.3 Espacios vectoriales euclídeos

Para llegar a la definición de *espacio normado* queda por introducir el último ingrediente necesario, la **norma**.

Producto escalar

Definición 2. Sea V un \mathbb{R} -espacio vectorial, un **producto escalar** en V es una aplicación

$$\langle , \rangle : V \times V \longrightarrow \mathbb{R}$$

verificando las siguientes propiedades:

1. $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle, \quad \forall \mathbf{u}, \mathbf{v} \in V.$
2. $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle, \quad \forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V.$
3. $\langle a\mathbf{u}, \mathbf{v} \rangle = a \langle \mathbf{u}, \mathbf{v} \rangle, \quad \forall \mathbf{u}, \mathbf{v} \in V.$
4. $\forall \mathbf{u} \in V, \langle \mathbf{u}, \mathbf{u} \rangle \geq 0$ y $\langle \mathbf{u}, \mathbf{u} \rangle = 0 \iff \mathbf{u} = 0.$

Definición 3. Se define como **espacio vectorial euclídeo** aquel espacio vectorial real V que tiene un producto escalar definido en él (V, \langle , \rangle) .

Si consideramos dos vectores $\mathbf{u}, \mathbf{v} \in V$, vistas como matriz fila y matriz columna respectivamente, el producto matricial define un producto escalar.

Demostración. 1. $\mathbf{u}\mathbf{v} = \sum_i^n u_i v_i = \sum_i^n v_i u_i = \mathbf{v}^T \mathbf{u}^T$

2. Inmediato por la propiedad 4. del producto de matrices

3. Inmediato por la propiedad 5. del producto de matrices.

4. $\mathbf{v}^T \mathbf{v} = \sum_i^n v_i v_i = \sum_i^n v_i^2$. Como $v_i^2 \geq 0, \quad \forall i \in \{1, \dots, n\} \Rightarrow \mathbf{v}^T \mathbf{v} \geq 0.$

La igualdad se da cuando:

$$v_i^2 = 0, \quad \forall i \in \{1, \dots, n\} \iff v_i = 0, \quad \forall i \in \{1, \dots, n\} \iff \mathbf{v} = 0$$

□

Norma de un vector

Sea $(V, \langle \cdot, \cdot \rangle)$ un espacio vectorial euclídeo, se define la **norma** de un vector $\mathbf{v} \in V$ por:

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$$

(considerando la determinación positiva de la raíz cuadrada).

Las propiedades que se obtienen de manera inmediata con la definición de norma son las siguientes:

1. $\|\mathbf{v}\| \geq 0$.
2. $\|\mathbf{v}\| = 0 \iff \mathbf{v} = 0$.
3. $\|a\mathbf{v}\| = |a| \|\mathbf{v}\|$.

Con el producto matricial se tendría la siguiente norma:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

Definición 4. Se define como **espacio vectorial normado** aquel espacio vectorial V que tiene asociada una norma. En el caso de que la norma venga dada por un producto escalar, este espacio normado también será un espacio euclídeo.

Resultados

Se obtienen dos resultados importantes:

Teorema 2 (Desigualdad de Schwartz). *Sea $(V, \langle \cdot, \cdot \rangle)$ un espacio vectorial euclídeo. Para cada $\mathbf{x}, \mathbf{y} \in V$ se verifica:*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\|$$

Demostración. Si $\mathbf{y} = 0$, el resultado es claro. Se supone por tanto que $\mathbf{y} \neq 0$. De la definición de producto escalar se obtiene que, para todo $\lambda \in \mathbb{R}$, se tiene:

$$\langle \mathbf{x} - \lambda \mathbf{y}, \mathbf{x} - \lambda \mathbf{y} \rangle \geq 0$$

En consecuencia:

$$\langle \mathbf{x}, \mathbf{x} \rangle - 2\lambda \langle \mathbf{x}, \mathbf{y} \rangle + \lambda^2 \langle \mathbf{y}, \mathbf{y} \rangle \geq 0$$

Ahora, tomando $\lambda = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{y}, \mathbf{y} \rangle}$, se tiene:

$$\langle \mathbf{x}, \mathbf{x} \rangle - 2 \frac{\langle \mathbf{x}, \mathbf{y} \rangle^2}{\langle \mathbf{y}, \mathbf{y} \rangle} + \frac{\langle \mathbf{x}, \mathbf{y} \rangle^2}{\langle \mathbf{y}, \mathbf{y} \rangle} \geq 0$$

Es decir:

$$\|\mathbf{x}\|^2 - \frac{\langle \mathbf{x}, \mathbf{y} \rangle^2}{\|\mathbf{y}\|^2} \geq 0$$

Multiplicando por $\|\mathbf{y}\|^2$:

$$\|\mathbf{x}\|^2 \|\mathbf{y}\|^2 - \langle \mathbf{x}, \mathbf{y} \rangle^2 \geq 0$$

De donde:

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 \leq \|\mathbf{x}\|^2 \|\mathbf{y}\|^2$$

Y finalmente:

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\|$$

□

Teorema 3 (Desigualdad Triangular). *Sea $(V, \langle \cdot, \cdot \rangle)$ un espacio vectorial euclídeo. Para cada $\mathbf{x}, \mathbf{y} \in V$ se verifica:*

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$$

Demostración. Usando la [Desigualdad de Schwartz](#):

$$\begin{aligned} \|\mathbf{x} + \mathbf{y}\|^2 &= \langle \mathbf{x} + \mathbf{y}, \mathbf{x} + \mathbf{y} \rangle = \\ &= \langle \mathbf{x}, \mathbf{x} \rangle + 2 \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle = \|\mathbf{x}\|^2 + 2 \langle \mathbf{x}, \mathbf{y} \rangle + \|\mathbf{y}\|^2 = \\ &\leq \|\mathbf{x}\|^2 + \|\mathbf{x}\| \|\mathbf{y}\| + \|\mathbf{y}\|^2 = \\ &\leq (\|\mathbf{x}\| + \|\mathbf{y}\|)^2 \end{aligned}$$

Así pues:

$$\|\mathbf{x} + \mathbf{y}\|^2 \leq (\|\mathbf{x}\| + \|\mathbf{y}\|)^2$$

y siendo ambos términos de la desigualdad positivos, se obtiene:

$$\|\mathbf{x} + \mathbf{y}\| \leq (\|\mathbf{x}\| + \|\mathbf{y}\|)$$

□

1.1.3 Aplicaciones lineales

Las aplicaciones lineales desempeñan en el estudio de los espacios vectoriales el mismo papel que las aplicaciones en el estudio de los conjuntos. Siendo los espacios vectoriales conjuntos dotados de una estructura adicional (las operaciones suma y producto por escalares) nos interesarán las aplicaciones que trasladen esta estructura de uno a otro espacio vectorial.

Definición 5. Dados dos espacios vectoriales V, V' , se dice que una aplicación

$$f : V \longrightarrow V'$$

es una **aplicación lineal** si verifica:

1. $f(\mathbf{u} + \mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v}), \quad \forall \mathbf{u}, \mathbf{v} \in V.$
2. $f(a\mathbf{u}) = af(\mathbf{u}), \quad \forall a \in \mathbb{R}, \quad \forall \mathbf{u} \in V.$

Lema 2. Una aplicación $f : V \longrightarrow V'$ entre dos espacios vectoriales, es lineal si, y solamente si,

$$f(a\mathbf{u} + b\mathbf{v}) = af(\mathbf{u}) + bf(\mathbf{v}), \quad \forall \mathbf{u}, \mathbf{v} \in V$$

Demostración. Suponiendo que f es lineal, entonces $f(a\mathbf{u} + b\mathbf{v}) = f(a\mathbf{u}) + f(b\mathbf{v})$ aplicando el punto 1 de la definición. Aplicando ahora el punto 2 de la definición se tiene que $f(a\mathbf{u}) + f(b\mathbf{v}) = af(\mathbf{u}) + bf(\mathbf{v})$

Recíprocamente, si $f(a\mathbf{u} + b\mathbf{v}) = af(\mathbf{u}) + bf(\mathbf{v}), \quad \forall \mathbf{u}, \mathbf{v} \in V$ si hacemos $a = 1, b = 1$ tenemos el punto 1 de la definición. Si hacemos $a = 1, b = 0$, tenemos el punto 2 de la definición, y por lo tanto f sería lineal. \square

1.2 DIFERENCIABILIDAD

El aprendizaje de los modelos o la capacidad de extraer información que explique decisiones de una red convolucional utilizan la *diferenciabilidad* para conocer las variaciones de los parámetros. Es una herramienta que ayuda a interpretar los resultados obtenidos por las redes, y actuar en base a dichas interpretaciones. Así, en esta sección, con la ayuda de [7] se va a iniciar el estudio del cálculo diferencial para funciones de varias variables en un contexto muy general, definiendo la diferenciabilidad de funciones entre espacios normados arbitrarios. Siendo X, Y dos espacios normados arbitrarios con $X \neq \{0\}$.

1.2.1 Definición de función diferenciable

Definición 6. Siendo $f : A \longrightarrow Y$ una función definida en $A \subseteq X$, se dice que f es **diferenciable** en un punto interior $\mathbf{a} \in A$ cuando existe una aplicación lineal y continua $g : X \longrightarrow Y$ verificando una de las siguientes condiciones equivalentes:

1. $\lim_{x \rightarrow a} \frac{\|f(x) - f(a) - g(x-a)\|}{\|x-a\|} = 0$
2. $\lim_{x \rightarrow a} \frac{f(x) - f(a) - g(x-a)}{\|x-a\|} = 0$

1.2.2 Propiedades de la diferencial

Se van a comentar algunas propiedades de la definición de función diferenciable.

Unicidad de la diferencial

Lema 3. Si la función f es diferenciable a entonces, la aplicación $g : X \rightarrow Y$ es única (notación de la Definición 6).

Demostración. Fijando $\mathbf{v} \in X$ vector no nulo, con el cambio $\mathbf{x} = \mathbf{a} + t\mathbf{v}$, $t \in \mathbb{R}^+$, se tiene que:

$$\begin{aligned} 0 &= \lim_{t \rightarrow 0} \frac{\|f(\mathbf{a} + t\mathbf{v}) - f(\mathbf{a}) - g((\mathbf{a} + t\mathbf{v}) - \mathbf{a})\|}{\|(\mathbf{a} + t\mathbf{v}) - \mathbf{a}\|} \\ &= \lim_{t \rightarrow 0} \frac{\|f(\mathbf{a} + t\mathbf{v}) - f(\mathbf{a}) - tg(\mathbf{v})\|}{|t| \|\mathbf{v}\|} \\ &= \frac{1}{\|\mathbf{v}\|} \lim_{t \rightarrow 0} \left\| \frac{f(\mathbf{a} + t\mathbf{v}) - f(\mathbf{a})}{t} - g(\mathbf{v}) \right\| \end{aligned}$$

Donde se ha usado la linealidad de g . Así:

$$g(\mathbf{v}) = \lim_{t \rightarrow 0} \frac{f(\mathbf{a} + t\mathbf{v}) - f(\mathbf{a})}{t}$$

Luego, si dos aplicaciones lineales $g_1, g_2 : X \rightarrow Y$ verifican la Definición 6, entonces $g_1 = g_2$. \square

Definición 7. La aplicación lineal g de la Definición 6 es conocida como la **diferencial** de f en \mathbf{a} , y se denota por $Df(\mathbf{a})$. Además su expresión es:

$$Df(\mathbf{a})(\mathbf{v}) = \lim_{t \rightarrow 0} \frac{f(\mathbf{a} + t\mathbf{v}) - f(\mathbf{a})}{t} \quad (1)$$

Relación con la continuidad

Lema 4. Si f es diferenciable en \mathbf{a} , entonces f es continua en \mathbf{a} .

Demostración. Con la siguiente igualdad:

$$f(\mathbf{x}) = f(\mathbf{a}) + \underbrace{Df(\mathbf{a})(\mathbf{x} - \mathbf{a})}_{\xrightarrow{\mathbf{x} \rightarrow \mathbf{a}} 0} + \underbrace{\|\mathbf{x} - \mathbf{a}\| \frac{f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})}{\|\mathbf{x} - \mathbf{a}\|}}_{\xrightarrow{\mathbf{x} \rightarrow \mathbf{a}} 0} \xrightarrow{\mathbf{x} \rightarrow \mathbf{a}} f(\mathbf{a})$$

Y esto se cumple para todo \mathbf{x} en un entorno de \mathbf{a} , con lo que tenemos que es continua en \mathbf{a} .

□

1.2.2.1 Independencia de las normas

La diferenciabilidad de f en a , así como su diferencial $Df(a)$, se conservan al sustituir las normas de X e Y por otras equivalentes³ a ellas. Denotando por $\|\cdot\|_1$ a las normas de partida de X e Y , y por $\|\cdot\|_2$ otras equivalentes a ellas, existen constantes $\lambda, \rho \in \mathbb{R}^+$, tales que $\lambda\|\mathbf{x}\|_1 \leq \|\mathbf{x}\|_2$ para todo $\mathbf{x} \in X$, y $\|\mathbf{y}\|_2 \leq \rho\|\mathbf{y}\|_1$ para todo $\mathbf{y} \in Y$. Suponiendo que f es diferenciable en \mathbf{a} con las normas $\|\cdot\|_1$, para $\mathbf{x} \in X$ interior no nulo, se tiene:

$$0 \leq \frac{\|f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\|_2}{\|\mathbf{x} - \mathbf{a}\|_2} \leq \frac{\rho \|f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\|_1}{\lambda \|\mathbf{x} - \mathbf{a}\|_1}$$

Y se deduce que $Df(\mathbf{a})$ sigue siendo la diferencial de f en \mathbf{a} para las normas $\|\cdot\|_2$. Como en espacios de dimensión finita las normas son equivalentes, cuando se trabaje con estos espacios no habrá que preocuparse por la norma utilizada.

1.2.3 Regla de la cadena

Se introduce a continuación un resultado clave para el cálculo de diferenciales.

Teorema 4 (Regla de la cadena). *Sean X, Y, Z tres espacios normados, Ω_X, Ω_Y subconjuntos abiertos de X e Y respectivamente, y sean $f : \Omega_X \rightarrow \Omega_Y$ y $g : \Omega_Y \rightarrow Z$ dos funciones. Si f es diferenciable en un punto $\mathbf{a} \in \Omega_X$ y g es diferenciable en $\mathbf{b} = f(\mathbf{a})$, entonces la composición $(g \circ f)$ es diferenciable en \mathbf{a} , con:*

$$D(f \circ g)(\mathbf{a}) = Dg(\mathbf{b}) \circ Df(\mathbf{a}) = Dg(f(\mathbf{a})) \circ Df(\mathbf{a})$$

Demostración. Se divide la demostración en tres fases:

Primera fase

Se define la función $\Phi : \Omega_X \rightarrow \mathbb{R}_0^+$ de la siguiente forma:

$$\Phi(\mathbf{x}) = \frac{\|f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\|}{\|\mathbf{x} - \mathbf{a}\|}, \quad \forall \mathbf{x} \in (\Omega_X - \{\mathbf{a}\}) \quad y \quad \Phi(\mathbf{a}) = 0$$

³ Dos normas $\|\cdot\|_1, \|\cdot\|_2$ en un espacio vectorial X son equivalentes si, y sólo si, existen constantes $\lambda, \rho \in \mathbb{R}^+$ tales que $\lambda\|\mathbf{x}\|_1 \leq \|\mathbf{x}\|_2 \leq \rho\|\mathbf{x}\|_1, \quad \forall \mathbf{x} \in X$

Por ser f diferenciable en \mathbf{a} , se tiene que Φ es continua en \mathbf{a} y verifica:

$$\Phi(\mathbf{x})\|\mathbf{x} - \mathbf{a}\| = \|f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\|, \quad \forall \mathbf{x} \in \Omega_X$$

Análogamente se define $\Psi : \Omega_Y \rightarrow \mathbb{R}_0^+$:

$$\Psi(\mathbf{y}) = \frac{\|g(\mathbf{y}) - g(\mathbf{b}) - Dg(\mathbf{b})(\mathbf{y} - \mathbf{b})\|}{\|\mathbf{y} - \mathbf{b}\|}, \quad \forall \mathbf{y} \in (\Omega_Y - \{\mathbf{b}\}) \quad y \quad \Psi(\mathbf{b}) = 0$$

Entonces Ψ es continua en \mathbf{b} y verifica

$$\Psi(\mathbf{y})\|\mathbf{y} - \mathbf{b}\| = \|g(\mathbf{y}) - g(\mathbf{b}) - Dg(\mathbf{b})(\mathbf{y} - \mathbf{b})\|, \quad \forall \mathbf{y} \in \Omega_Y$$

Con la finalidad de abbreviar la notación, se define también $\Lambda : \Omega_X \rightarrow \mathbb{R}_0^+$ por

$$\Lambda(\mathbf{x}) = \|(g \circ f)(\mathbf{x}) - (g \circ f)(\mathbf{a}) - (Dg(\mathbf{b}) \circ Df(\mathbf{a}))(\mathbf{x} - \mathbf{a})\|, \quad \forall \mathbf{x} \in \Omega_X$$

Ya que $Df(\mathbf{b}) \circ Df(\mathbf{a})$ es una aplicación lineal entre X y Z , bastará ver que

$$\lim_{x \rightarrow \mathbf{a}} \frac{\Lambda(\mathbf{x})}{\|\mathbf{x} - \mathbf{a}\|} = 0$$

Segunda fase

Se obtiene ahora una desigualdad clave, que relaciona Λ con Φ y Ψ . Para $\mathbf{x} \in \Omega_X$, se tiene $\mathbf{y} = f(\mathbf{x}) \in \Omega_Y$, y se usa en Z la [desigualdad triangular](#):

$$0 \leq \Lambda(\mathbf{x}) \leq \|g(\mathbf{y}) - g(\mathbf{b}) - Dg(\mathbf{b})(\mathbf{y} - \mathbf{b})\| + \|Dg(\mathbf{b})(\mathbf{y} - \mathbf{b} - Df(\mathbf{a})(\mathbf{x} - \mathbf{a}))\| \quad (2)$$

Ahora, trabajando con los dos sumandos que han aparecido en esta última expresión:

$$\begin{aligned} \|Dg(\mathbf{b})(\mathbf{y} - \mathbf{b} - Df(\mathbf{a})(\mathbf{x} - \mathbf{a}))\| &\leq \|Dg(\mathbf{b})\| \|\mathbf{y} - \mathbf{b} - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\| \\ &= \|Dg(\mathbf{b})\| \|f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\| \\ &= \|Dg(\mathbf{b})\| \Phi(\mathbf{x}) \|\mathbf{x} - \mathbf{a}\| \end{aligned} \quad (3)$$

El otro sumando de (2):

$$\|g(\mathbf{y}) - g(\mathbf{b}) - Dg(\mathbf{b})(\mathbf{y} - \mathbf{b})\| = \Psi(\mathbf{y})\|\mathbf{y} - \mathbf{b}\| = \Psi(f(\mathbf{x}))\|f(\mathbf{x}) - f(\mathbf{a})\|$$

La desigualdad triangular en Y nos da:

$$\begin{aligned} \|f(\mathbf{x}) - f(\mathbf{a})\| &\leq \|f(\mathbf{x}) - f(\mathbf{a}) - Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\| + \|Df(\mathbf{a})(\mathbf{x} - \mathbf{a})\| \\ &\leq (\Phi(\mathbf{x}) + \|Df(\mathbf{a})\|) \|\mathbf{x} - \mathbf{a}\| \end{aligned}$$

De donde se deduce que:

$$\|g(\mathbf{y}) - g(\mathbf{b}) - Dg(\mathbf{b})(\mathbf{y} - \mathbf{b})\| \leq \Psi(f(\mathbf{x}))(\Phi(\mathbf{x}) + \|Df(\mathbf{a})\|) \|\mathbf{x} - \mathbf{a}\| \quad (4)$$

Usando (3) y (4) se deduce de (2) la desigualdad que se buscaba:

$$0 \leq \Lambda(\mathbf{x}) \leq \Psi(f(\mathbf{x})) (\Phi(\mathbf{x}) + \|Df(\mathbf{a})\|) \|\mathbf{x} - \mathbf{a}\| + \|Dg(\mathbf{b})\| \Phi(\mathbf{x}) \|\mathbf{x} - \mathbf{a}\|$$

Tercera fase

Para $\mathbf{x} \in (\Omega_X - \{\mathbf{a}\})$, de la última desigualdad se deduce:

$$0 \leq \frac{\Lambda(\mathbf{x})}{\|\mathbf{x} - \mathbf{a}\|} \leq \Psi(f(\mathbf{x})) (\Phi(\mathbf{x}) + \|Df(\mathbf{a})\|) + \|Dg(\mathbf{b})\| \Phi(\mathbf{x})$$

Luego bastará probar que el último miembro de esta desigualdad tiene límite 0 en el punto \mathbf{a} .

Puesto que f es continua en el punto \mathbf{a} y Ψ es continua en el punto $\mathbf{b} = f(\mathbf{a})$, se ve que $\Psi \circ f$ es continua en \mathbf{a} . Como Φ también es continua en \mathbf{a} , se tiene:

1. $\lim_{\mathbf{x} \rightarrow \mathbf{a}} \Psi(f(\mathbf{x})) = \Psi(f(\mathbf{a})) = \Psi(\mathbf{b}) = 0$
2. $\lim_{\mathbf{x} \rightarrow \mathbf{a}} \Phi(\mathbf{x}) = \Phi(\mathbf{a}) = 0$

Así, se concluye que:

$$\lim_{\mathbf{x} \rightarrow \mathbf{a}} (\Psi(f(\mathbf{x})) (\Phi(\mathbf{x}) + \|Df(\mathbf{a})\|) + \|Dg(\mathbf{b})\| \Phi(\mathbf{x})) = 0$$

Esto implica que $\lim_{\mathbf{x} \rightarrow \mathbf{a}} \frac{\Lambda(\mathbf{x})}{\|\mathbf{x} - \mathbf{a}\|} = 0$ como se quería demostrar.

□

La regla de la cadena es la herramienta fundamental del algoritmo de **backpropagation**. Este algoritmo es indispensable para el aprendizaje de redes neuronales, ya que permite (como su nombre indica) *propagar* la información para *atrás* (desde la salida hacia la entrada) de manera eficiente. La potencia de **backpropagation** reside fundamentalmente en esta eficiencia: reutilizando las derivadas de capas posteriores se pueden ir calculando las derivadas de manera recursiva.

1.2.4 Vector Gradiente

Como caso particular se estudiará la diferenciabilidad cuando el espacio de salida es \mathbb{R}^N con $N > 1$ y el de llegada es \mathbb{R} para llegar al concepto de *vector gradiente*.

Definición 8. Si $\mathbf{u} \in X = \mathbb{R}^N$ y $\|\mathbf{u}\| = 1$, se dice que f es **derivable en la dirección \mathbf{u}** , en el punto \mathbf{a} (notación de la Definición 6), cuando la función $\phi_{\mathbf{u}} :]-r, r[\rightarrow Y = \mathbb{R}$ definida por $\phi_{\mathbf{u}}(t) = f(\mathbf{a} + t\mathbf{u})$ es derivable en el punto \mathbf{a} y en la dirección \mathbf{u} , que se denota por $f'_{\mathbf{u}}(\mathbf{a})$:

$$f'_{\mathbf{u}}(\mathbf{a}) = \phi'_{\mathbf{u}}(0) = \lim_{t \rightarrow 0} \frac{f(\mathbf{a} + t\mathbf{u}) - f(\mathbf{a})}{t} \in Y$$

Además se pueden sacar dos ideas de la definición:

1. f es derivable en la dirección $\mathbf{u} \iff f$ es derivable en la dirección $-\mathbf{u}$.
2. Si f es diferenciable en \mathbf{a} , entonces f es direccionalmente derivable en \mathbf{a} y, para todo $\mathbf{u} \in X$ con $\|\mathbf{u}\| = 1$, se tiene que

$$f'_{\mathbf{u}}(\mathbf{a}) = Df(\mathbf{a})(\mathbf{u})$$

1.2.4.1 Derivadas parciales

Cogiendo la base usual de X , $\{e_1, e_2, \dots, e_N\}$ y fijando un $k \in \{1, \dots, N\}$ definimos:

Definición 9. Si f es derivable en la dirección e_k en el punto \mathbf{a} , se dice que f es **parcialmente derivable con respecto a la k -ésima variable** en el punto \mathbf{a} . Entonces, la derivada direccional de f en \mathbf{a} en la dirección e_k se denomina **derivada parcial de f con respecto a la k -ésima variable** en el punto \mathbf{a} , y se denota por $\frac{\partial f}{\partial x_k}(\mathbf{a})$, es decir:

$$\frac{\partial f}{\partial x_k}(\mathbf{a}) = f'_{e_k}(\mathbf{a}) = \lim_{t \rightarrow 0} \frac{f(\mathbf{a} + t\mathbf{e}_k) - f(\mathbf{a})}{t}$$

Si esto ocurre para todo $k \in \{1, \dots, N\}$ se dice que f es **parcialmente derivable** en \mathbf{a} , y se tienen N derivadas parciales $\frac{\partial f}{\partial x_k}(\mathbf{a})$, $k \in \{1, \dots, N\}$.

Definición 10. Si f es parcialmente derivable en \mathbf{a} , se define el **gradiente** de f en \mathbf{a} como:

$$\nabla f(\mathbf{a}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{a}), \frac{\partial f}{\partial x_2}(\mathbf{a}), \dots, \frac{\partial f}{\partial x_N}(\mathbf{a}) \right) = \sum_{k=1}^N \frac{\partial f}{\partial x_k}(\mathbf{a}) e_k$$

1.3 OPTIMIZACIÓN

Para abordar el problema de la optimización, las derivadas serán esenciales ya que dan información útil sobre cómo varía la función con pequeños cambios en su dominio. La derivada (o gradiente), se puede ver como un vector que apunta hacia la dirección donde la función crece más rápido. Utilizando esta información, parece fácil pensar que si la función se mueve en dirección contraria al gradiente se conseguirá un proceso de minimización de la función. Esto es precisamente lo que hace la técnica de **descenso de Gradiente**.

1.3.1 Descenso de Gradiente

El *descenso de gradiente* es un algoritmo de optimización iterativo. Se tiene una función $f : D \subset \mathbb{R}^N \rightarrow \mathbb{R}$ es diferenciable en un entorno de un punto \mathbf{a} , entonces f disminuye más rápido si va desde el punto \mathbf{a} en dirección opuesta al gradiente de f en \mathbf{a} : $-\nabla f(\mathbf{a})$. Si $\mathbf{x}_0 = \mathbf{a}$, entonces se puede calcular de manera iterativa \mathbf{x}_n , $n \in \mathbb{N}$:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \nabla f(\mathbf{x}_n)$$

Para controlar la velocidad de cambio, se añade un parámetro denominado *learning rate* $\eta \in \mathbb{R}^+$. El *learning rate* es un parámetro muy relevante ya que permite escapar del mínimo local al que se dirige el método simplemente alterando su valor. Para funciones complejas, donde el cálculo del mínimo no es viable hacerlo de forma analítica, el *learning rate* juega un papel fundamental. [10]

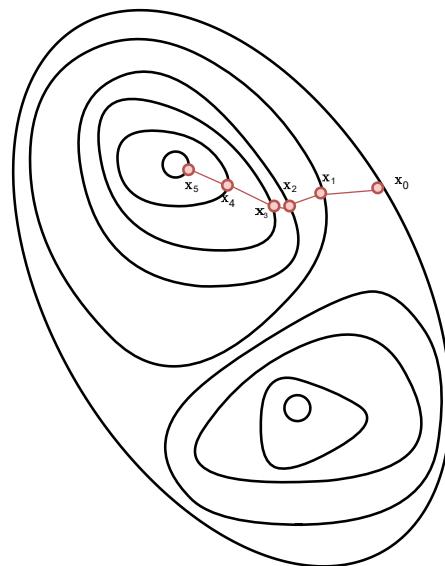


Figura 1: Ilustración de ejemplo para el descenso por gradiente

1.4 PROBABILIDAD

Algunas ramas de la informática como la Inteligencia Artificial lidian con problemas donde los sucesos no son deterministas, y por tanto se tiene que manejar la incertidumbre. La *probabilidad* es una rama de las matemáticas que aporta herramientas para trabajar con esta incertidumbre de forma rigurosa. Con ayuda de Ash [8] se introducirán las bases que necesitaremos para el aprendizaje automático.

1.4.1 Fundamentos de la probabilidad

Definición 11. Un *espacio de probabilidad* es un concepto que sirve para modelar un cierto experimento aleatorio. Concretamente, un **espacio de probabilidad** es una terma (Ω, \mathcal{A}, P) donde:

- Ω es un conjunto arbitrario.
- $\mathcal{A} \subset \Omega$ tiene estructura de σ -álgebra. Para ello tiene que verificar:
 1. $\mathcal{A} \neq \emptyset$.
 2. \mathcal{A} es cerrada para las uniones numerables.
 3. \mathcal{A} es cerrada para la formación del complementario.

Los elementos de \mathcal{A} se llaman **sucesos**.

- $P : \mathcal{A} \longrightarrow [0, 1]$ es una **función de probabilidad**. Para ello tiene que satisfacer los siguientes axiomas:
 1. $P(A) \geq 0, \quad \forall A \in \mathcal{A}$.
 2. $P(\Omega) = 1$.
 3. Para cualquier secuencia $\{A_n\}_{n \in \mathbb{N}} \subseteq \mathcal{A}$ de sucesos disjuntos se tiene que:

$$P\left(\bigcup_{n \in \mathbb{N}} A_n\right) = \sum_{n \in \mathbb{N}} P(A_n)$$

Es interesante estudiar la probabilidad de un suceso restringiéndose a que haya ocurrido alguna condición previa. Así, se puede estudiar la probabilidad de un determinado suceso condicionada a otro.

Definición 12. Sea (Ω, \mathcal{A}, P) un espacio de probabilidad y $A \in \mathcal{A}$ un suceso tal que $P(A) > 0$. La aplicación

$$P(\cdot / A) : \mathcal{A} \longrightarrow [0, 1]$$

$$P(B / A) = \frac{P(B \cap A)}{P(A)}$$

se llama **probabilidad condicionada**. Al espacio $(\Omega, \mathcal{A}, P(\cdot / A))$ se llama *espacio de probabilidad condicionada al suceso A*.

Calcular la probabilidad de ocurrencia de un suceso condicionado a otro implica estudiar la relación entre dos sucesos. Estos sucesos pueden estar relacionados fuertemente, lo que intuitivamente haría que la probabilidad condicionada de uno

condicionado al otro se viese muy afectada a la probabilidad que habría sin la condición. Esta relación fuerte se puede ver como una *dependencia alta* entre sucesos. Así, se introduce la definición de **independencia**.

Definición 13. Sea (Ω, \mathcal{A}, P) un espacio de probabilidad. Dos sucesos $A, B \in \mathcal{A}$ se dirán que son **independientes** cuando

$$P(A \cap B) = P(A)P(B)$$

Ahora, surge el interés de definir una función que asigne valores reales a cada elemento de un espacio muestral para que así, los conjuntos de interés estén definidos en términos de esa función. De esta forma, surge el concepto de **variable aleatoria**.

Definición 14. Una **variable aleatoria** sobre un espacio de probabilidad (Ω, \mathcal{A}, P) es una función medible

$$X : \Omega \longrightarrow \mathbb{R}$$

tal que

$$X^{-1}(B) = \{\omega \in \Omega : X(\omega) \in B\}, \quad B \in \mathcal{B}$$

Siendo \mathcal{B} la σ -álgebra de Borel ⁴

Ahora, se está en condición de definir la manera en el que se calculan las probabilidades en una variable aleatoria.

Definición 15. Si X es una variable aleatoria sobre (Ω, \mathcal{A}, P) , se define su **distribución de probabilidad** como una función

$$\mathbb{P}_X : \mathcal{B} \longrightarrow [0, 1]$$

definida por:

$$\mathbb{P}_X(B) = P(\{\omega \in \Omega; X(\omega) \in B\}) = P(X \in B), \quad \forall B \in \mathcal{B}$$

Definición 16. Se llama **función de distribución de la variable aleatoria** X a la función $F_X : \mathbb{R} \longrightarrow [0, 1]$ definida por:

$$F_X(x) = \mathbb{P}_X((-\infty, x]) = P(X \in (-\infty, x]), \quad \forall x \in \mathbb{R}$$

En función de la naturaleza de la variable aleatoria se pueden distinguir:

⁴ La σ -álgebra de Borel es la mínima σ -álgebra sobre \mathbb{R} que contiene a todos los intervalos.

- **Variable aleatoria discreta.** Se dice que $X : (\Omega, \mathcal{A}, P) \rightarrow (\mathbb{R}, \mathcal{B}, \mathbb{P}_X)$ es una variable aleatoria discreta si existe un conjunto numerable $E_X \subset \mathbb{R}$ tal que $\mathbb{P}_X(X \in E_X) = 1$.

Las variables aleatorias discretas poseen una **función masa de probabilidad** p_X donde

$$p_X : E_X \rightarrow [0, 1], \quad p_X(x) = P(X = x), \quad \forall x \in E_X$$

además, se cumple que $\sum_{x \in E_X} p_X(x) = 1$.

Se define su **esperanza matemática** como

$$\mathbb{E}[X] = \sum_{x \in E_X} x P(X = x)$$

siempre que dicha suma sea absolutamente convergente, es decir

$$\sum_{x \in E_X} |x| P(X = x) < \infty$$

- **Variable aleatoria continua.** Se dice que $X : (\Omega, \mathcal{A}, P) \rightarrow (\mathbb{R}, \mathcal{B}, \mathbb{P}_X)$ es una variable aleatoria continua si existe una función $f_X : \mathbb{R} \rightarrow \mathbb{R}$ tal que

$$F_X(x) = \int_{-\infty}^x f_X(y) dy, \quad \forall x \in \mathbb{R}$$

f_X recibe el nombre de función de densidad de X .

Se define su **esperanza matemática** como

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x) dx$$

siempre que dicha suma sea absolutamente convergente, es decir

$$\int_{\mathbb{R}} |x| f_X(x) dx < \infty$$

A la esperanza matemática se la conoce también como *valor esperado* o *media* de la variable aleatoria X .

1.4.2 Desigualdad de Hoeffding

Como resultado de los fundamentos de probabilidad vistos, se quiere llegar a un resultado esencial en el área del Aprendizaje Automático: la [Desigualdad de Hoeffding](#). Se verá en la parte de informática como esta desigualdad sustentará el concepto de aprendizaje, permitiendo asentar las bases y seguir con el estudio de herramientas más avanzadas de Aprendizaje Automático.

Se introducirá la [Desigualdad de Markov](#) y un lema previo para tener todas las herramientas necesarias para probar la [Desigualdad de Hoeffding](#).

Teorema 5 (desigualdad de Markov). *Sea X una variable aleatoria y $a > 0$, entonces*

$$P(|X| \geq a) \leq \frac{\mathbb{E}(|X|)}{a}$$

Demostración. Para cualquier evento o suceso A , sea $\mathbb{1}_A$ la función indicadora o característica de A : $\mathbb{1}_A = 1$ si ocurre A y $\mathbb{1}_A = 0$ en caso contrario.. Luego es claro que: $a\mathbb{1}_{|X| \geq a} \leq |X|$. Por lo tanto $\mathbb{E}(a\mathbb{1}_{|X| \geq a}) \leq \mathbb{E}(|X|)$. Como

$$a\mathbb{E}(\mathbb{1}_{|X| \geq a}) = aP(|X| \geq a),$$

se tiene que:

$$aP(|X| \geq a) \leq \mathbb{E}(|X|) \stackrel{a > 0}{\iff} P(|X| \geq a) \leq \frac{\mathbb{E}(|X|)}{a}$$

□

Lema 5. *Sea X una variable aleatoria con $\mathbb{E}[X] = 0$ y $X \in [a, b]$. Entonces, para todo $\lambda > 0$,*

$$\mathbb{E}(\exp(\lambda X)) \leq \exp(\lambda^2(b-a)^2/8)$$

Demostración. Por la convexidad de la función exponencial,

$$\exp(\lambda x) \leq \frac{x-a}{b-a}\exp(\lambda b) + \frac{b-x}{b-a}\exp(\lambda a), \quad a \leq x \leq b$$

Por lo tanto,

$$\begin{aligned} \mathbb{E}[\exp(\lambda X)] &\leq \mathbb{E}\left[\frac{X-a}{b-a}\right]\exp(\lambda b) + \mathbb{E}\left[\frac{b-X}{b-a}\right]\exp(\lambda a) \\ &= \frac{b}{b-a}\exp(\lambda a) - \frac{a}{b-a}\exp(\lambda b), \quad (\mathbb{E}[X] = 0) \\ &= (1 - \theta + \theta \exp(\lambda(b-a))) \exp(-\theta\lambda(b-a)), \quad \theta = \frac{-a}{b-a} \end{aligned}$$

definiendo ahora $u = \lambda(b-a)$ y la función $\phi(u) = -\theta u + \log(1 - \theta + \theta e^u)$. Se tiene entonces que

$$\mathbb{E}[\exp(\lambda X)] \leq \exp(\phi(u))$$

Ahora bien, para minimizar la cota superior se va a expresar $\phi(u)$ en una serie de Taylor⁵ con residuo:

$$\phi(u) = \phi(0) + u\phi'(0) + \frac{u^2}{2}\phi''(\nu), \quad \nu \in [0, u]$$

Luego,

$$\begin{aligned} \phi'(u) &= -\theta + \frac{\theta e^u}{1 - \theta + \theta e^u} \Rightarrow \phi'(0) = 0 \\ \phi''(u) &= \frac{\theta e^u}{1 - \theta + \theta e^u} - \frac{(\theta e^u)^2}{(1 - \theta + \theta e^u)^2} \\ &= \frac{\theta e^u}{1 - \theta + \theta e^u} \left(1 - \frac{\theta e^u}{1 - \theta + \theta e^u}\right) \\ &= \rho(1 - \rho) \end{aligned}$$

Ahora bien, $\phi''(u)$ es maximizado cuando

$$\rho = \frac{\theta e^u}{1 - \theta + \theta e^u} = \frac{1}{2} \Rightarrow \phi''(u) \leq \frac{1}{4}$$

Por lo tanto,

$$\phi(u) \leq \frac{u^2}{8} = \frac{\lambda^2(b-a)^2}{8}$$

Luego,

$$\mathbb{E}[\exp(\lambda X)] \leq \exp\left(\frac{\lambda^2(b-a)^2}{8}\right)$$

□

Teorema 6 (Desigualdad de Hoeffding). *Sean X_1, \dots, X_n variables aleatorias independientes tales que $X_i \in [a_i, b_i]$, donde $S_n = \sum_{i=1}^n X_i$. Entonces, se tiene que:*

$$P(|S_n - \mathbb{E}[S_n]| \geq \eta) \leq 2\exp\left(-2\frac{\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right), \quad \eta > 0$$

5 Una serie de Taylor de una función f es una aproximación de funciones mediante una serie de potencias o suma de potencias enteras de polinomios como $(x - a)^n$ llamados términos de la serie. Dicha suma se calcula a partir de las derivadas de la función f para un determinado valor o punto a suficientemente derivable sobre la función y un entorno sobre el cual converja la serie.

$$Taylor(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}}{n!} (x - a)^n$$

Demostración. Para $\lambda, \eta > 0$, la [desigualdad de Markov](#), la independencia de los X_i y el [lema 5](#) conducirá al resultado.

$$\begin{aligned}
 P(S_n - \mathbb{E}[S_n] \geq \eta) &= P(e^{\lambda(S_n - \mathbb{E}[S_n])} \geq e^{\lambda\eta}) \leq && \text{Des. Markov} \\
 &\leq e^{-\lambda\eta} \mathbb{E}[e^{\lambda(S_n - \mathbb{E}[S_n])}] = && X_i \text{ indep.} \\
 &= e^{-\lambda\eta} \prod_{i=1}^n \mathbb{E}[e^{\lambda(X_i - \mathbb{E}[X_i])}] \leq && \text{lema 5} \\
 &\leq e^{-\lambda\eta} \prod_{i=1}^n e^{\frac{\lambda^2(b_i - a_i)}{8}} = \\
 &= \exp\left(-\lambda\eta + \sum_{i=0}^n \frac{\lambda^2(b_i - a_i)}{8}\right)
 \end{aligned}$$

Para obtener la mejor cota posible se puede encontrar el mínimo de la cota como función de λ , definimos $g(\lambda) = -\lambda\eta + \sum_{i=0}^n \frac{\lambda^2(b_i - a_i)}{8}$. Como g es una función cuadrática tiene mínimo, este mínimo es $\lambda^* = \frac{4\eta}{\sum(b_i - a_i)^2}$. Como la cota es $\exp(g(\lambda))$ y la exponencial es una función estrictamente creciente, se tiene que su mínimo también se alcanza en λ^* . Así, reemplazando en la expresión final tendríamos el resultado:

$$\mathbb{P}(|S_n - \mathbb{E}[S_n]| \geq \eta) \leq 2\exp\left(-2\frac{\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

□

Una aplicación de esta desigualdad se verá en la [sección 2.2.1.2](#) enfocada al Aprendizaje Automático.

Parte II

TEORÍA INFORMÁTICA

Inteligencia Artificial eXplicable (XAI) en imágenes, desde los fundamentos. Se hablará de conceptos clave: error, funciones de pérdida, criterio de aprendizaje (método de optimización), compromiso sesgo-varianza, regularización, tipos de problemas... De ahí se pasará a estudiar los modelos de Perceptrón[9] y Perceptrón multicapa[9] que darán pie al Aprendizaje Profundo [10] con las Redes Convolucionales. Teniendo ya las herramientas para resolver el problema se introducirá el concepto de la Inteligencia Artificial eXplicable junto con las técnicas que se van a usar: CAM, Grad-CAM, Grad-CAM++ y Smooth Grad-CAM++.

2

INTELIGENCIA ARTIFICIAL

2.1 INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

Marvin Minsky, matemático, informático y famoso especialista en Inteligencia Artificial (Toosi et al. [17]), define Inteligencia Artificial como:

“La ciencia que hace que las máquinas hagan cosas que requerirían de inteligencia si fueran hechas por un humano”.

McKinsey & Company, por su parte, la define enfatizando capacidades que requieren las máquinas (Toosi et al. [17]):

“La capacidad de las máquinas para imitar las funciones cognitivas humanas, incluidas la percepción, el razonamiento, el aprendizaje y la resolución de problemas”.

En 1950 Alan Turing publicó un artículo en 1950 (“Computers and intelligence”, Turing [18]) donde propone un método para determinar la diferencia entre una tarea realizada por una máquina y una tarea realizada por una persona. Este método es conocido como el *Test de Turing* y consiste en una serie de preguntas realizadas por un interrogador, el cual tendrá que determinar, en función de las respuestas, si han sido respondidas por una máquina o por un ser humano.

El campo de la Inteligencia Artificial ha experimentado subidas y caídas en cuanto a su relevancia a lo largo de los últimos 70 años. Un cuadro resumen de estos ciclos a lo largo de la historia se puede encontrar en la [Figura 2](#). En este resumen se ve a grandes rasgos como la IA pasa por distintas fases marcadas por su enfoque:

1. Algoritmos de Búsqueda
2. Sistemas Expertos
3. Machine Learning

4. Deep Learning

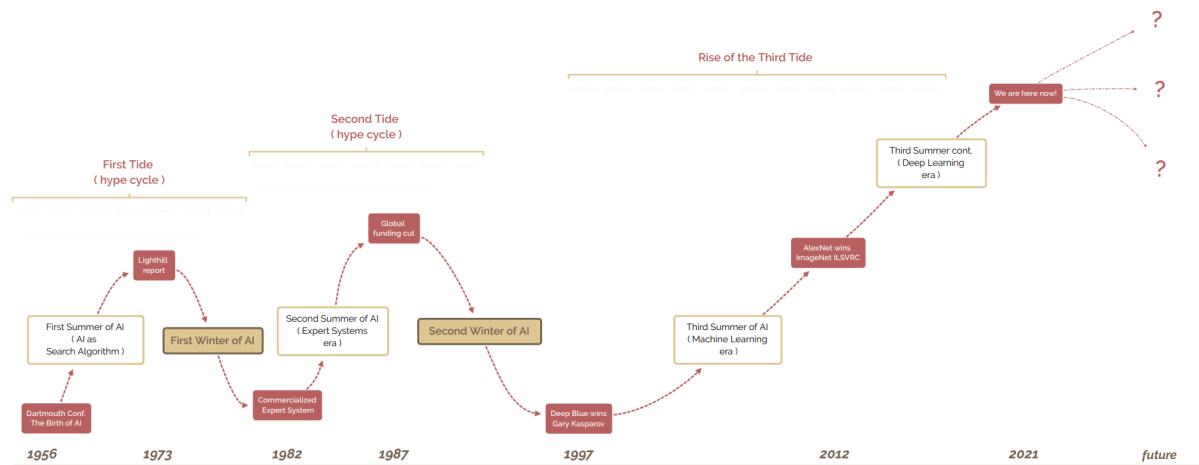


Figura 2: Ciclos de relevancia de la Inteligencia Artificial a lo largo de su historia. Reflejo de las subidas y caídas marcadas por sus fases. (Toosi et al. [17])

Actualmente el mundo se encuentra en la mejor época de la Inteligencia Artificial, en una pendiente alcista de relevancia marcada a su inicio con el **Aprendizaje Automático (Machine Learning)** y continuada hasta la fecha con el **Aprendizaje Profundo (Deep Learning)**. Estas nuevas áreas de la IA han abierto un gran abanico de posibilidades de interés de estudio científico que desarrollaremos en las siguientes secciones.

2.2 APRENDIZAJE AUTOMÁTICO

El **Aprendizaje Automático (AA)** es una rama de la *Inteligencia Artificial* que se centra en el uso de datos y algoritmos para asemejarse a los humanos en el aprendizaje, mejorando gradualmente su desempeño [19]. Cuando se habla de AA, se habla de la capacidad de mejora de una máquina a través de la experiencia en un determinado tipo de tarea. Su objetivo es automatizar la tarea de construir modelos analíticos a través del uso de algoritmos que, iterativamente, aprenden de datos específicos del problema. Esto permite a estas máquinas adquirir una visión fina del problema y encontrar patrones complejos sin necesidad de ser explícitamente programada (Janiesch et al. [20]).

Dado el problema y la disposición de datos, se pueden distinguir tres tipos de aprendizaje: supervisado, no supervisado y por refuerzo.

- *Aprendizaje supervisado* En este tipo, el proceso de generación de conocimiento se realiza a través de un conjunto de datos (input) etiquetados (output real). Así, los pares (input,output) sirven para calibrar los parámetros libres internos del modelo.

Una vez que el modelo está entrenado, ya está listo para procesar datos no vistos previamente y arrojar un resultado que aproximará mejor al output real cuanto mejor haya sido el entrenamiento previo. Generalmente, en este tipo de aprendizaje, podemos distinguir los problemas de *regresión* y *clasificación*. [20]

- *Aprendizaje no supervisado* es otro de los tipos de *Aprendizaje Automático*. En el cual se incluyen conjuntos de datos sin etiquetar en los que no se conoce previamente la estructura que estos poseen. En este tipo de aprendizaje se busca obtener información clave o importante sin la conocer previamente la referencia de las variables de salida, explorando la estructura de los datos que no están etiquetados. [21]
- *Aprendizaje por refuerzo* En los sistemas que implementan este tipo de aprendizaje, en vez de tener unos datos de entrada y sus salidas, se describe el estado actual del sistema, se especifica un objetivo y se le da al sistema una lista de acciones permitidas con las restricciones del entorno para los resultados de esas acciones. Con esta predisposición se le deja al modelo ganar experiencia alcanzando el objetivo a través del principio de prueba error, maximizando su recompensa. [20]

Dentro del *Aprendizaje Supervisado*, dependiendo del problema, se ofrecen varias alternativas: modelos de regresión, modelos bayesianos, redes neuronales, etc.

La familia de las *Redes Neuronales* adquiere un papel protagonista en este estudio. Esta familia está determinada por un modelo matemático formado por unidades de procesamiento, denominadas neuronas, distribuidas por capas que se transmiten la información entre ellas. Cada conexión entre neuronas genera unas señales que serán procesadas y transmitidas por las siguientes. Las *Redes Neuronales* suelen estar organizadas por capas de neuronas. Cuando el número de capas es alto, el campo de estudio pasa a denominarse *Aprendizaje Profundo (Deep Learning - DL)*.

El *DL* es realmente útil en problemas donde se manejan una enorme cantidad de datos con grandes dimensiones, donde el *AA* no es capaz de dar resultados competentes. El campo de estudio del *DL* suele ir asociado a problemas de visión por computador o procesado de lenguaje natural, por ejemplo.

En el resto de la sección 2.2 el estudio se centrará en el *Aprendizaje Supervisado*.

2.2.1 Fundamentos

El Aprendizaje Supervisado trata de resolver, por tanto, un problema en base a datos. Así, se puede representar el problema a través de una función objetivo *desconocida* f que dado un estado (datos) x , es capaz de dar la solución $f(x) = y$. Por lo tanto, se

aspira a encontrar una función g , a través de un algoritmo de aprendizaje \mathcal{A} , que sea capaz de aproximarse lo máximo posible a f .

Notación

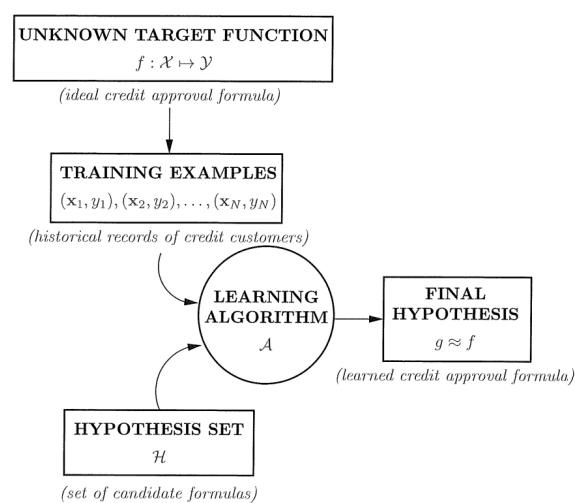


Figura 3: Esquema básico de un problema de aprendizaje automático. (Figure 1.2, Abu-Mostafa et al. [9])

- \mathcal{X} : es el espacio de entrada del problema (dominio). Las instancias se notarán con x .
- \mathcal{Y} : es el espacio de salida del problema (etiquetas). Las instancias se notarán con y .
- $f : \mathcal{X} \rightarrow \mathcal{Y}$ es la función objetivo desconocida.
- $\mathbb{P}(\mathcal{X})$: distribución de los datos
- \mathcal{D} : conjunto de datos del que se dispone para el problema. se tiene que

$$\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$$

y es finito. Esto es,

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}, \quad N \in \mathbb{N}$$

donde $f(x_n) = y_n$ para cada $n \in \{1, 2, \dots, N\}$
Los elementos de \mathcal{D} están independientemente e idénticamente distribuidos (i.i.d.)

- \mathcal{H} : clase de funciones o conjunto de hipótesis de donde se escogerá la función $g \in \mathcal{H}$ que aproxima a f . Tienen de dominio \mathcal{X} y

$$h(\mathcal{X}) \subset \mathcal{Y} \quad \forall h \in \mathcal{H}$$

la clase de funciones vendrá determinada por los pesos W que acompañarán a los datos x

- \mathcal{A} : algoritmo de aprendizaje

2.2.1.1 Regresión y clasificación

Estos son los dos problemas esenciales del AA. En un problema de regresión se buscará ajustar una salida **continua**. Por otra parte, en un problema de clasificación,

se buscará clasificar el objeto en clases, que no son más que conjuntos **discretos**. Así, en función del problema que se presente, se tendrán que tomar unas decisiones u otras para afrontar el problema.

Estos problemas son los que guiarán esta introducción al Aprendizaje Automático supervisado.

2.2.1.2 Bondad de la solución

Del aprendizaje no se espera obtener la función real f , sino aproximarla con g lo máximo posible. Para saber la bondad de esta aproximación necesitamos definir un **error de medida** ($E(h, f)$ donde $h \in \mathcal{H}$) que lo cuantifique. Nuestro objetivo por lo tanto es minimizar E , esto es:

- Encontrar $g^* \in \mathcal{H}$ tal que

$$E(g^*, f) = \min_{h \in \mathcal{H}} E(h, f)$$

Para este proceso de minimización se tiene que utilizar algún algoritmo de aprendizaje \mathcal{A} . Este algoritmo de aprendizaje será escogido en función de la naturaleza del problema.

Volviendo al error, hay que destacar que el dominio de E es un par de funciones definidas, cada una, sobre todo el espacio \mathcal{X} . Sin embargo, no se dispone de la información de cómo se comporta f en todo \mathcal{X} , solo se tiene una muestra $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$ para trabajar con f . Por lo tanto, solo se puede hablar del error dentro de \mathcal{D} , el **error dentro de la muestra**, que se notará como

$$E_{in}(h) = E_{in}(h, f)$$

y que se nombrará como E_{in} cuando se hable de g , es decir, $E_{in} = E_{in}(g)$. De manera análoga llamaremos al **error fuera de la muestra** E_{out} . Como las funciones de \mathcal{H} vienen determinadas por sus pesos w , se puede llamar también a los errores con $E_{in}(w)$ y $E_{out}(w)$.

En este punto, surge una pregunta: ¿Se puede concluir algo de E_{out} a través de E_{in} ? En otras palabras, ¿es posible generalizar E_{in} para todo el espacio muestral? Pues la respuesta, adelantando acontecimientos, es que con E_{in} se puede acotar E_{out} .

Desigualdad de Hoeffding

La desigualdad de Hoeffding (sección 1.4.2, Teorema 6) viene para acabar con la duda planteada. Considerando la situación de $\mathcal{H} = \{h\}$, con solo una función.

$Z_i = [[f(x_i) \neq h(x_i)]]$ ¹ representa una nueva variable. Para cualquier muestra \mathcal{D} de tamaño N , se define

- $E_{in}^{\mathcal{D}}(h) = \frac{1}{N} \underbrace{\sum_{i=1}^N ([[f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]])}_{S_N^{\mathcal{D}}} = \frac{1}{N} S_N^{\mathcal{D}}$
- $E_{out}(h) = \mathbb{P}([[f(x) \neq h(x)]]) = \mathbb{E}_{\mathcal{D}}[E_{in}^{\mathcal{D}}(h)] = \mathbb{E}_{\mathcal{D}}[\frac{1}{N} S_N^{\mathcal{D}}] = \frac{1}{N} \mathbb{E}_{\mathcal{D}}[S_N^{\mathcal{D}}]$.

Entonces

$$\begin{aligned} \mathbb{P}(\mathcal{D} : |E_{in}(h) - E_{out}(h)| \geq \varepsilon) &= \mathbb{P}(\mathcal{D} : \frac{1}{N} |S_N^{\mathcal{D}} - \mathbb{E}[S_N^{\mathcal{D}}]| \geq \varepsilon) \\ &= \mathbb{P}(\mathcal{D} : |S_N^{\mathcal{D}} - \mathbb{E}[S_N^{\mathcal{D}}]| \geq N\varepsilon) \quad \text{Des. Hoeffding} \\ &\leq 2e^{-\frac{2(N\varepsilon)^2}{N}} \\ &\leq 2e^{-2\varepsilon^2 N} \quad \forall \varepsilon > 0 \end{aligned}$$

Se ha podido aplicar la **Desigualdad de Hoeffding** porque Z_1, Z_2, \dots, Z_N son i.i.d. y además $Z_i \in \{0, 1\}, \quad \forall i \in \{1, 2, \dots, N\}$.

Reescribiendo el resultado:

$$\mathbb{P}(\mathcal{D} : |E_{in}(h) - E_{out}(h)| \leq \varepsilon) \geq 1 - 2e^{-2\varepsilon^2 N} \quad \forall \varepsilon > 0 \quad (5)$$

La clave de este resultado está en que la cota depende del tamaño de la muestra \mathcal{D} que se disponga. En la [ecuación \(5\)](#) se aprecia como a medida que crece N , la probabilidad de que E_{in} esté cerca de E_{out} aumenta.

Cuando tenemos una clase de funciones \mathcal{H} con más de un elemento, la [ecuación \(5\)](#) se reescribiría como:

$$\mathbb{P}(\mathcal{D} : |E_{in}(g) - E_{out}(g)| \leq \varepsilon) \geq 1 - 2|\mathcal{H}|e^{-2\varepsilon^2 N} \quad \forall \varepsilon > 0 \quad (6)$$

Si se nota como $\delta = 2|\mathcal{H}|e^{-2\varepsilon^2 N}$, tenemos que, con probabilidad de al menos $1 - \delta$:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2|\mathcal{H}|}{\delta}}$$

Por lo tanto, se puede afirmar que un modelo aprende a través de una muestra:

1. $E_{in} \approx E_{out}$ (**Desigualdad de Hoeffding**)
2. $E_{in} \approx 0$ (algoritmo de aprendizaje \mathcal{A})

¹ NOTACIÓN. $[[f(x_i) \neq h(x_i)]] = |\{x_i : f(x_i) \neq h(x_i)\}|$

Ambas condiciones juntas dan como resultado $E_{out} \approx 0$, el objetivo final.

Y es precisamente, con el tamaño de \mathcal{H} de donde surge la necesidad de usar un conjunto de **test**. Este conjunto será un subconjunto de \mathcal{D} *no visto antes por el modelo* y que será evaluado por el mismo cuando todos sus parámetros ya estén fijos, es decir, cuando ya hayan sido entrenados y se tengan los valores finales. Esto implica que el conjunto *test* solo será visto por $\mathcal{H}_{test} = \{g\}$ que contiene una única función, haciendo una mejor aproximación a lo que sería el error fuera de la muestra. Obtendríamos una mejor cota de E_{out} .

2.2.1.3 Entrenamiento

El entrenamiento es la fase del problema donde se construye el modelo. Para el entrenamiento surge el interés de usar un conjunto de datos que simule el conjunto de datos *test* para poder seleccionar hiperparámetros en el aprendizaje y tener una estimación real de la bondad del modelo durante el aprendizaje. Este conjunto de datos se llamará **validación**. Además, para llevar a cabo el aprendizaje, es necesario escoger un criterio de aprendizaje y una función de pérdida.

2.2.1.4 Función de pérdida

La función de pérdida guiará al modelo a coger los mejores pesos para el problema. Así, el objetivo de esta función es minimizarla por lo que la pregunta que hay que hacerse es: ¿qué se quiere minimizar para poder tener la mejor solución?

- La función de pérdida será referenciada como ℓ cuando se hable de un método general y no se diga explícitamente cómo se calcula el error.

Error Cuadrático Medio

El Error Cuadrático Medio (MSE) es usado en problemas de regresión, donde se busca ajustar un valor continuo. El MSE mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima.

$$MSE = \mathbb{E}[(f(\mathbf{x}) - h(\mathbf{x}))^2] = \frac{1}{N} \sum_{n=1}^N (f(\mathbf{x}_n) - h(\mathbf{x}_n))^2$$

El MSE es el segundo momento (sobre el origen) del error, y por lo tanto **incorpora tanto la varianza del estimador como su sesgo** (7). En una analogía con la desviación estándar, tomando la raíz cuadrada del MSE produce el error de la raíz cuadrada de

la media o la desviación de la raíz cuadrada media (RMSE o RMSD), que tiene las mismas unidades que la cantidad que se estima. [10]

$$\begin{aligned}
 MSE &= \mathbb{E}[(f(\mathbf{x}) - h(\mathbf{x}))^2] \\
 &= \mathbb{E}[((f(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})]) - (h(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})]))^2] \\
 &= \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])^2] + \mathbb{E}[(h(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])^2] \\
 &\quad - 2\mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])(h(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])] \\
 &= \underbrace{(f(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])^2}_{bias} + \underbrace{\mathbb{E}[(h(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])^2]}_{varianza} \\
 &\quad - 2(f(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])\mathbb{E}[h(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})]] \\
 &= bias^2 + varianza - 2(f(\mathbf{x}) - \mathbb{E}[h(\mathbf{x})])\left(\underbrace{\mathbb{E}[h(\mathbf{x})] - \mathbb{E}[h(\mathbf{x})]}_0\right) \\
 &= bias^2 + varianza
 \end{aligned} \tag{7}$$

Entropía cruzada

Error usado para problemas de clasificación. Este error se basa en las probabilidades de que un elemento de la muestra \mathbf{x} pertenezca a una clase. Para el caso binario, $\mathcal{Y} = \{0, 1\}$. Cada instancia \mathbf{x} con etiqueta y , tendrá una probabilidad p de pertenecer a la clase $y = 1$. Entonces, el error de entropía cruzada binaria se puede escribir como

$$BCE = -(y \log(p) + (1 - y) \log(1 - p))$$

Cuando tenemos varias instancias se puede calcular como la media de estos errores.

$$BCE = -\frac{1}{N} \sum_{n=1}^N (y_n \log(p_n) + (1 - y_n) \log(1 - p_n))$$

Para el caso multiclase con C clases, q será la probabilidad de que x pertenezca a su clase y (para el caso binario: $q = p$ si $y = 1$, $q = 1 - p$ si $y \neq 1$). Entonces

$$CE = -\sum_{n=1}^C y_n \log(q_n)$$

2.2.1.5 Criterio de aprendizaje - SGD

A partir de las funciones de pérdida se obtiene información sobre cómo de bien está ajustando el modelo a los datos que se le han proporcionado como entrada. Con esta información hay que ser capaz de modificar el modelo para acercarlo a la función verdadera. Es decir, ir buscando $h \in \mathcal{H}$ que se vayan aproximando cada vez más a f . Esto se hace minimizando el error. Para minimizar el error existen varios métodos, como por ejemplo el descenso de gradiente [1.3.1](#):

$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j} \quad (8)$$

donde se utilizan todos los datos de la muestra dentro de E_{in} . Esto puede provocar problemas de estancamiento, ya que el gradiente se calculará usando siempre todas las muestras y llegará un punto en el que las variaciones sean mínimas. Como solución a ello surge el **Descenso de Gradiente Estocástico (SGD)**. Consiste en tener en cuenta únicamente $M < N$ datos aleatorios. Esto influye en el error que se va a calcular, ya que solo tendrá cuenta esos M datos. [Algoritmo \[1\]](#)

Algoritmo 1: Gradiente Descendente Estocástico

```

1 Inicializar  $\mathbf{w} \leftarrow \mathbf{w}_0$ ,  $\eta \leftarrow \eta_0$ ,  $\varepsilon \leftarrow \varepsilon_0$ 
2 while  $E_{in} > \varepsilon$  do
3   desordenar y dividir la muestra en mini – batches
4   for minibatch  $\in$  mini – batches do
5      $w_j := w_j - \eta \frac{\partial E_{in}^{minibatch}(\mathbf{w})}{\partial w_j}$ 

```

2.2.1.6 Compromiso sesgo-varianza

Con el [MSE](#) (ecuación [\(7\)](#)) se ha visto como el error se puede descomponer en *sesgo* y *varianza*. Cuando un modelo es lo suficientemente complejo para poder ajustarse mucho a los datos, la varianza va a ser muy alta. Así, el error estará marcado precisamente por esa varianza, ese sobreajuste. En la otra cara de la moneda están los modelos demasiado simples, que tienen un error alto, esta vez debido al sesgo. Surge por lo tanto la necesidad de buscar un compromiso *sesgo-varianza*.

El tamaño de \mathcal{H} marcará la complejidad del modelo: cuanto más grande sea el conjunto \mathcal{H} mayor será la complejidad del modelo y, por tanto, mejor se podrá ajustar a los datos durante el entrenamiento. Tanto es así, que sería posible ajustarse al 100 % a los datos de \mathcal{D} , pero como se ha visto en la [Desigualdad de Hoeffding](#), cuanto más complejo sea el conjunto \mathcal{H} peor se acotará E_{out} . Esto lleva a tener un control sobre el conjunto \mathcal{H} :

1. Simplificar la clase de funciones \mathcal{H} .
2. Aplicar técnicas para controlar el ajuste y establecer restricciones a la hora de buscar la función objetivo. Esta opción es más conocida como **regularización**.

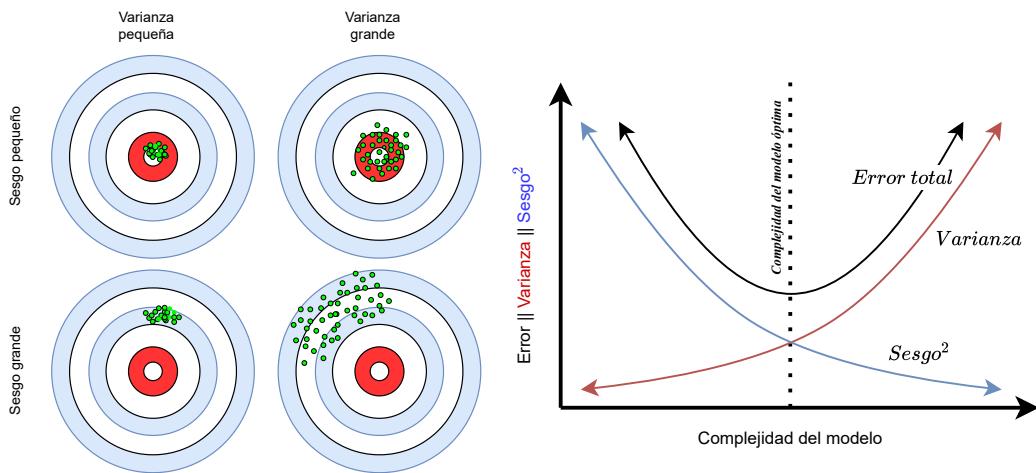


Figura 4: Ejemplo básico de la compensación sesgo-varianza acompañado de una gráfica que muestra como la complejidad del modelo afecta de manera directa a esta compensación reflejándolo, en última instancia, en el error.

Al entrenar el modelo hay que intentar no sobreajustarlo a los datos \mathcal{D} . En la Figura 4 se ve como a mayor complejidad mayor varianza, y a menor complejidad mayor sesgo. Hay que encontrar el equilibrio entre ambas expresiones. La **regularización** ayudará a no sobreajustarnos a los datos. Algunos métodos de regularización consisten en, por ejemplo, añadir una penalización a la función de pérdida ℓ . En otros modelos más complejos como las redes neuronales existe el **Dropout** que se verá más adelante.

2.2.2 Perceptrón

Una vez introducidos los conceptos genéricos del aprendizaje automático, el estudio se va a centrar en modelos concretos que van a ser necesarios para la experimentación posterior. Un ejemplo de modelo simple del *Aprendizaje Supervisado* es el *Perceptrón* [9]. El **Perceptrón**, es uno de los primeros modelos de aprendizaje supervisado en redes neuronales y consiste en ajustar los pesos y el parámetro de polarización mediante ajustes pequeños.

En la Figura 5 puede verse un modelo de perceptrón donde podemos distinguir las distintas entradas x_1, x_2, \dots, x_m , sus pesos w_1, w_2, \dots, w_m junto al sesgo w_0 , y la aplicación de la función de activación f .

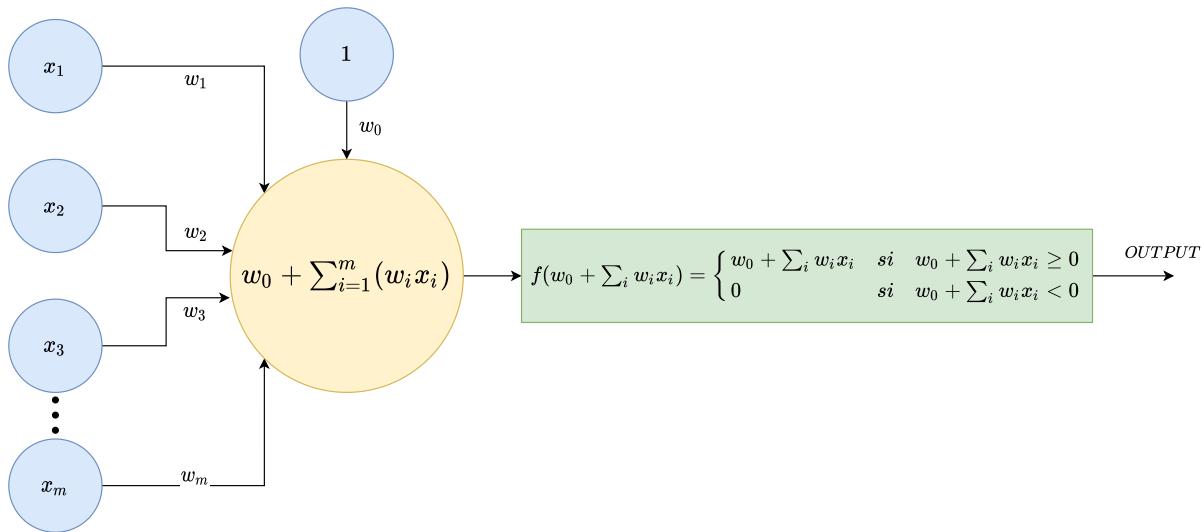


Figura 5: Resumen gráfico del funcionamiento de una neurona o perceptrón.

A la *función de activación* f utilizada en la Figura 5 se le denomina **Rectified Linear Unit (ReLU)**. Estas capas de activación juegan un papel relevante dentro del modelo, ya que incorporan la no linealidad al modelo. De otro modo, estos modelos serían simples modelos lineales.

Algoritmo 2: Perceptron Learning Algoritmo (PLA)

```

1 Inicializar  $\mathbf{w} \leftarrow \mathbf{w}_0$ 
2 while Haya cambios do
3   for  $x_i \in \mathcal{D}$  do
4     if  $\text{sign}(\mathbf{w}^T \mathbf{x}_i) \neq y_i$  then
5        $\mathbf{w} = \mathbf{w} + y_i \mathbf{x}_i$ 

```

La actualización de los pesos tratan de corregir los errores uno a uno. Este algoritmo iterará de forma infinita si los datos no son linealmente separables.

2.2.3 Perceptrón Multicapa - Red Neuronal

La evolución del perceptrón es el *Perceptrón Multicapa (MultiLayer Perceptron - MLP)*, también conocido como el tipo básico de **Red Neuronal**. El MLP se basa en incluir varias unidades de perceptrones distribuidos por capas donde se comunican **todos** los perceptrones de una capa con **todos** los de la siguiente (*fully connected layer*), haciendo un paso unidireccional de los datos (*feedforward*).

Las capas adicionales del MLP se llaman *capas ocultas* y van a permitir definir modelos más complejos. Además tenemos la *capa de entrada* y la *capa de salida* que ya

teníamos en el Perceptrón simple. Cada capa está definida por el número de neuronas (unidades de activación, perceptrones simples). Así, el número de capas y el número de neuronas que tiene cada capa definen la arquitectura del MLP. Fijar la arquitectura equivale a fijar nuestra clase de funciones \mathcal{H} . Una vez se haya fijado la arquitectura, es momento de estimar los parámetros.(ver Figura 6)

Para denominar a la *función de activación* se utilizará la letra θ .

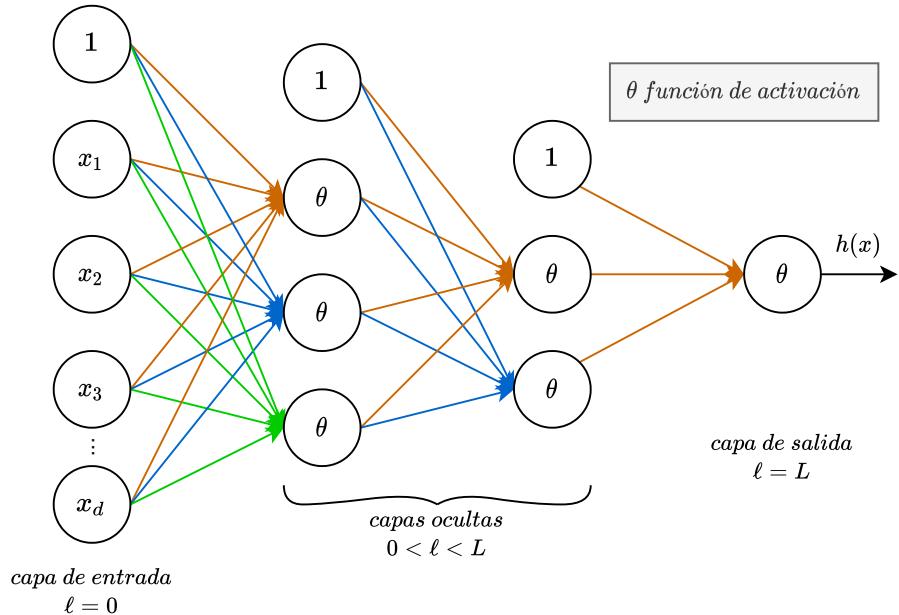


Figura 6: Esquema gráfico de un Perceptrón Multicapa con dos capas ocultas.

Notación

- Las capas son nombradas por $\ell = 0, 1, \dots, L$. En la Figura 6, $L = 3$.
 - *Capa de entrada* $\ell = 0$. No se considera una capa normalmente.
 - *Capa de salida* $\ell = L$. Determina el valor arrojado por el modelo.
 - *Capas ocultas* $0 < \ell < L$.
- Cada capa ℓ tiene dimensión $d(\ell)$, lo cual significa que tiene $d(\ell) + 1$ nodos. Siendo el nodo 0 el nodo del bias o sesgo que proporcionará siempre el valor 1 como dato y su peso determinará el valor del sesgo.
- Una hipótesis $h \in \mathcal{H}$ es seleccionada una vez se han fijado unos pesos.
- *Vector de entrada* a una capa ℓ , $s^{(\ell)}$. Tiene dimensión $d(\ell)$.
- *Vector de salida* de una capa ℓ , $x^{(\ell)}$. Tiene dimensión $d(\ell) + 1$.

- *Pesos de entrada a una capa ℓ , $W^{(\ell)}$.* Tiene dimensión $(d(\ell - 1) + 1) \times d(\ell)$.
- *Pesos de salida de una capa ℓ , $W^{(\ell+1)}$.* Tiene dimensión $(d(\ell) + 1) \times d(\ell + 1)$.

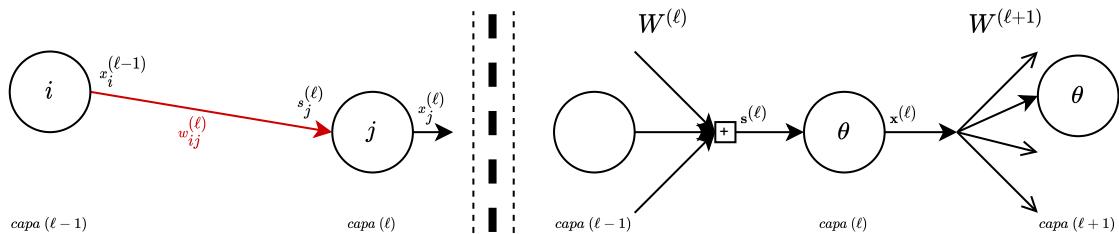


Figura 7: Notación utilizada

2.2.3.1 Paso de los datos a través del MLP

En la introducción al Perceptrón Multicapa se había mencionado que los datos pasan de una capa a la siguiente $\ell = 0 \rightarrow \ell = 1 \rightarrow \dots \rightarrow \ell = L$. ([Algoritmo 3](#))

Algoritmo 3: Forward Propagation

```

1  $x^{(0)} \leftarrow \mathbf{x}$ 
2 for  $\ell \in \{1, \dots, L\}$  do
3    $s^{(\ell)} \leftarrow (W^{(\ell)})^T x^{(\ell-1)}$ 
4    $x^{(\ell)} \leftarrow \begin{bmatrix} 1 \\ \theta(s^{(\ell)}) \end{bmatrix}$ 
5 return  $x^{(L)}$ 

```

Sin embargo, para actualizar los parámetros también será necesario hacer una propagación desde la salida hasta las primeras capas. Esto se hace a través de un algoritmo denominado *Backpropagation*.

Backpropagation

Backpropagation [10] es un algoritmo que calcula el gradiente de manera eficiente, está basado en la [regla de la cadena](#) para escribir las derivadas parciales de la capa ℓ usando las derivadas parciales de la capa $\ell + 1$. La actualización de los pesos requiere calcular $\nabla E_{in}(w)$. Para ello, es necesario conocer

$$\frac{\partial E_{in}}{\partial W^{(\ell)}} = \frac{1}{N} \sum_{n=1}^N \frac{\partial e_n(h(\mathbf{x}_n), y_n)}{\partial W^{(\ell)}}$$

Donde e_n es el error del modelo en el dato x_n .

Se denotará como $\delta^{(\ell)} = \frac{\partial e}{\partial s^{(\ell)}}$ al gradiente del error con respecto la señal de entrada de la capa ℓ . Entonces:

$$\frac{\partial e}{\partial W^{(\ell)}} = \frac{\partial e}{\partial s^{(\ell)}} \frac{\partial s^{(\ell)}}{\partial W^{(\ell)}} \underbrace{\begin{bmatrix} s^{(\ell)} \\ \frac{\partial s^{(\ell)}}{\partial W^{(\ell)}} \end{bmatrix}}_{\equiv} \underbrace{\begin{bmatrix} (W^{(\ell)})^T x^{(\ell-1)} \\ x^{(\ell-1)} \end{bmatrix}}_{x^{(\ell-1)} [\delta^{(\ell)}]^T}$$

Por otro lado como

$$\delta^{(\ell)} = \frac{\partial e}{\partial s^{(\ell)}} = \frac{\partial e}{\partial x^{(\ell)}} \frac{\partial x^{(\ell)}}{\partial s^{(\ell)}} = \underbrace{\frac{\partial e}{\partial s^{(\ell+1)}}}_{\frac{\partial e}{\partial x^{(\ell)}}} \underbrace{\frac{\partial s^{(\ell+1)}}{\partial x^{(\ell)}}}_{\theta'(s^{(\ell)})} \underbrace{\frac{\partial x^{(\ell)}}{\partial s^{(\ell)}}}_{W^{(\ell+1)} \delta^{(\ell+1)}}$$

y teniendo en cuenta que:

$$\delta^{(L)} = \frac{\partial e}{\partial s^{(L)}} = \frac{\partial e}{\partial x^{(L)}} \frac{\partial x^{(L)}}{\partial s^{(L)}} = \frac{\partial e}{\partial x^{(L)}} \theta'(s^{(L)})$$

donde se dispone de ambos términos para el producto (por ejemplo, para el caso del MSE, $\delta^{(L)} = 2(x^{(L)} - y)\theta'(s^{(L)})$). Así, se podrían calcular todos los $\delta^{(\ell)}$ de manera secuencial: $\delta^{(1)} \leftarrow \delta^{(2)} \leftarrow \dots \leftarrow \delta^{(L)}$

Algoritmo 4: Backpropagation: cálculo de los $\delta^{(\ell)}$

- 1 Ejecutar el [Algoritmo 3](#) de **Forward propagation**
- 2 Calcular $\frac{\partial e}{\partial x^{(L)}}$
- 3 Calcular $\theta'(s^{(L)})$
- 4 $\delta^{(L)} \leftarrow \frac{\partial e}{\partial x^{(L)}} \theta'(s^{(L)})$
- 5 **for** $\ell \in \{L-1, L-2, \dots, 1\}$ **do**
- 6 Calcular $\theta'(s^{(\ell)})$
- 7 $\delta^{(\ell)} \leftarrow \theta'(s^{(\ell)}) [W^{(\ell+1)} \delta^{(\ell+1)}]$

Con el [Algoritmo 4](#) ya se podría calcular el gradiente del error para aplicar el Descenso por Gradiente.

Dropout

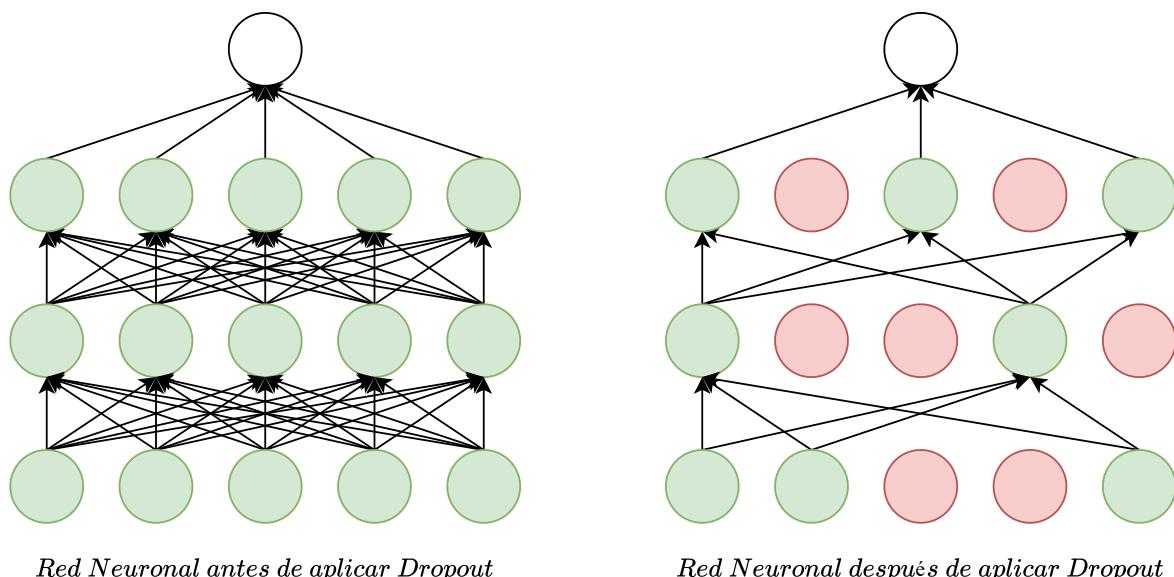
Las *Redes Neuronales* tienen un gran poder de aproximación a cualquier función. Por lo tanto, tienden a ajustarse a los datos de entrada y es por ello por lo que

surgen técnicas de regularización propias de las *Redes Neuronales*. Una de las técnicas de regularización muy usadas por las *Redes Neuronales* es el **Dropout**, técnica que protege del overfitting.

El **Dropout** hace que durante el entrenamiento, algunas neuronas de las capas son ignoradas (*dropped out*). Esto provoca que cada capa se comporte como si tuviese menos unidades, simulando, a nivel global, otra arquitectura de la red. Así, cada actualización de los pesos se hará, sobre una “arquitectura distinta”.

Este comportamiento se refleja en que el trabajo, o los patrones detectados, son repartidos de forma más igualitaria entre las neuronas, haciendo que las neuronas no se especialicen fuertemente en tareas muy concretas.

El “apagado” de las neuronas se hace a través de una probabilidad p fijada. Una vez se ha entrenado la *Red Neuronal*, el *Dropout* deja de tener efecto para el testeo o evaluación de nuevas muestras.



Red Neuronal antes de aplicar Dropout

Red Neuronal después de aplicar Dropout

Figura 8: Ejemplo gráfico de dropout cuando la probabilidad de apagado es $p = 0,5$

2.2.4 Deep Learning

El aprendizaje profundo surge de llevar estas *Redes Neuronales* un paso más allá. Una *Red Neuronal* con una capa es suficiente para representar cualquier función, lo que se llama *aproximador universal*², pero la capa debe ser increíblemente grande y puede dar problemas en el aprendizaje y en generalizar correctamente.

² Una red puede llegar a aproximar cualquier función tanto como se quiera modificando su arquitectura

Se vio cómo aumentando el número de capas (la profundidad), se podían reducir el tamaño de las capas de manera que se obtuviesen mismos resultados. Lo interesante era que con el aumento de profundidad se podían reducir el número de parámetros totales en la red. De aquí surge el interés por modelos profundos, que pertenecen ya al ámbito del **Aprendizaje Profundo o Deep Learning (DL)**. Existen muchas variantes de *Redes Neuronales Profundas* como las *Redes Convolucionales (CNNs)*.

2.3 REDES CONVOLUCIONALES

En la sección anterior se acabó hablando de cómo surge el interés en modelos de *Deep Learning* como las **Redes Convolucionales (CNNs)**. De aquí en adelante se asumirá que la entrada a la CNN será una imagen de dos dimensiones con tres canales (RGB).

El nombre de *Red Neuronal Convolucional* indica que la red utiliza una operación matemática llamada **convolución**. Convolución es un tipo especial de operación lineal. Así, las *Redes Convolucionales* no son más que redes neuronales que usan convolución en al menos una de sus capas. [10]

CONVOLUCIÓN [10] *En la forma más general, convolución es una operación de dos funciones reales. Sin embargo, cuando se trabaja con datos en un ordenador, en este caso con imágenes, los datos son discretos. Dado que cada elemento de la entrada y del núcleo debe almacenarse explícitamente por separado, se suele suponer que estas funciones son nulas en todas partes menos en el conjunto finito de puntos para los que se almacenan los valores. Esto significa que, en la práctica, se puede implementar la suma infinita como una suma sobre un número finito de elementos de la matriz.*

De esta manera, cogiendo la imagen bidimensional I de entrada como primera aplicación y una segunda aplicación K (matriz bidimensional) que se nombrará como kernel, se define la convolución como:

$$\begin{aligned} S(i, j) &= (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \\ &= (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \end{aligned}$$

Al aplicar la operación de convolución en las capas en lugar de aplicar capas totalmente conectadas, reducimos el número de conexiones, disminuyendo en gran medida el número total de parámetros.

2.3.1 Arquitectura de una Red Convolucional

Las Redes Convolucionales tienen como entrada imágenes, lo que obliga a diseñar una arquitectura que nos permita trabajar con ellas. En las capas ocultas, las neuronas que conforman una capa tienen tres dimensiones: *ancho, alto, profundo*.

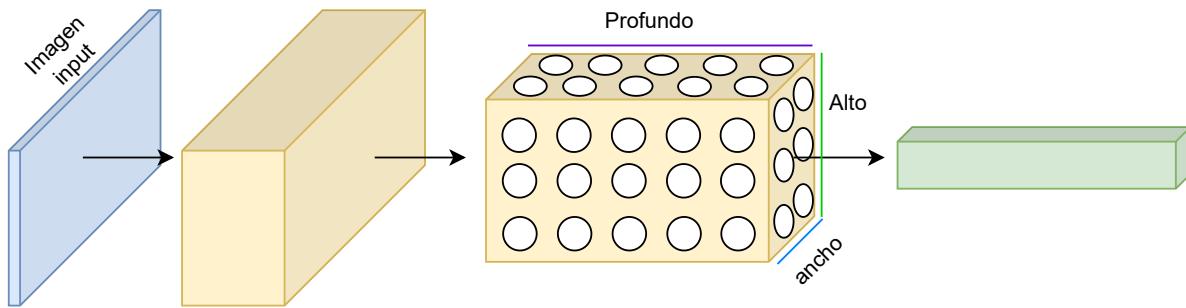


Figura 9: Red Convolucional donde se muestra como las neuronas se organizan en tres dimensiones (ancho, alto y profundo). Todas las capas de una Red Convolucional transforma una entrada de tres dimensiones en una salida de tres dimensiones. En este caso la entrada es una imagen de profundidad tres: los canales RGB.

Entonces, una capa de una Red Convolucional debe transformar un volumen de tres dimensiones en otro volumen de tres dimensiones. Las capas utilizadas para hacer estas transformaciones son, entre otras: capa convolucional, capa de pooling, capa totalmente conectada (*fully connected*, como en MLP), capa de activación...

Alguna de estas capas contienen parámetros y otras no: en particular, las capas convolucionales y las FC (*fully connected*) tienen parámetros, las capas de activación y pooling no tienen parámetros, y su función está predefinida y se mantendrán invariantes durante toda la vida del modelo.

Las Redes Convolucionales suelen tener dos partes esenciales en su arquitectura: el *extractor de características*, que está formado esencialmente por capas convolucionales y el *clasificador*, donde son las capas totalmente conectadas las que forman esta parte. [22]

2.3.1.1 Capa Convolucional

En una capa convolucional, los parámetros consisten en un conjunto de *kernels* o *filtros* que se van a aprender durante el entrenamiento. Estos filtros, son filtros de tres dimensiones, con un tamaño espacial (ancho y alto) inferior al volumen de entrada, pero con exactamente la misma profundidad. Durante el paso de los datos a través de la red hacia delante, cada filtro se convoluciona con el volumen de entrada, dando lugar a un mapa de activación de dos dimensiones. Intuitivamente la red aprenderá filtros que se activen cuando encuentren alguna característica visual (desde un borde,

hasta a una cara de una persona). Aplicando distintos filtros (de mismo tamaño) se construye un nuevo volumen de salida. [22]

El hecho de que el tamaño espacial de los filtros pueda ser menor que el tamaño espacial del volumen de entrada provoca que no todas las neuronas de una capa se conecten con todas las neuronas de la capa siguiente, como pasaba con las capas totalmente conectadas (FC). Se puede hablar, por tanto, de una *conexión local* entre capas, en donde cada neurona solo ve una región local espacial del volumen de entrada. A esta región local se le llamará **campo receptivo** de la neurona con respecto a una capa. [22]

Por otro lado, las dimensiones del volumen de salida depende de la *profundidad* del filtro, del *stride* (mínima distancia entre dos neuronas de la misma profundidad del campo receptivo de la capa anterior) y *padding* (rellenar con ceros cuando el filtro pase por los bordes del volumen). [22]

2.3.1.2 Capa de Pooling

Es común que entre capas de convolución aparezca una capa de pooling. Su función es disminuir progresivamente el tamaño espacial de los datos que pasan por la red para reducir el número de parámetros y para controlar el overfitting. Esta capa actúa sobre el tamaño espacial del volumen, dejando intacta la profundidad. Se puede ver como un filtro que en lugar de producir como salida una función lineal, produce una función diferente. Hay dos tipos esenciales de pooling: *MaxPooling* y *AveragePooling*.

- *MaxPooling*: coge el máximo valor de cada región local del volumen por el que pasa.
- *AveragePooling*: coge la media de todos los valores de cada región local por el que pasa.

Para el cálculo de los gradientes durante el backpropagation, en el caso del MaxPooling, normalmente se guarda el índice del valor más grande que se ha encontrado durante el paso hacia delante de los datos. [22]

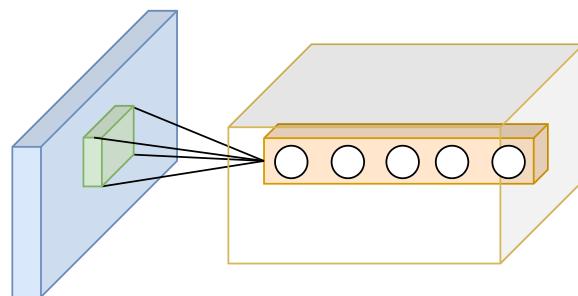


Figura 10: Campo receptivo de una serie de neuronas con misma posición espacial con respecto la capa anterior. (basado en [22])

2.3.1.3 Capas completamente conectadas

La salida de la última capa convolucional es un conjunto de datos de tres dimensiones. Con estos datos espaciales calculados desde la entrada se tiene que dar una predicción. Para ello, la salida del extractor de características se transforma en un conjunto de datos de una dimensión para poder ser procesado por las capas *totalmente conectadas* (fully connected).

Estas capas *totalmente conectadas* son las encargadas de clasificar las características extraídas de la entrada, dando el resultado final del modelo para esos datos de entrada.

2.3.2 Clasificación en Redes Convolucionales

En un problema de clasificación, queremos asignarle a una imagen de entrada una clase dentro de un dominio de clases predefinido $\{c_1, c_2, \dots, c_k\}$. Para ello, la imagen pasará primero por la parte de *extracción de características* (la parte convolucional de la red), y estas características extraídas serán pasadas por el clasificador (la parte totalmente conectada de la red) que tendrá una salida Y de tamaño k . Diremos que la red ha clasificado la imagen en la clase C cuando

$$\begin{cases} C = c_i \\ i = \text{argmax}(Y) \end{cases}$$

Normalmente, la salida Y es pasada por una capa *softmax*³ que devuelve unas probabilidades de pertenencia a la clase.

2.3.3 Detección de objetos

La **detección de objetos** es una técnica de visión por computador que aborda el problema de *identificar* y *localizar* los objetos dentro de una imagen o vídeo. La salida esperada de los modelos que se usan para detectar objetos es una lista de *bounding boxes* junto con sus etiquetas de clase, con las que se señalan regiones de la imagen ([Figura 11](#)).

³ $\text{softmax} : \mathbb{R}^K \rightarrow [0, 1]^K$, $\text{softmax}(Y)_{c_i} = \frac{e^{Y_{c_i}}}{\sum_{k=1}^K e^{Y_{c_k}}}$

Las primeras aproximaciones usadas para resolver este problema consistían en histogramas de color o de bordes, que sacaban ciertas características de la imagen con las que un modelo de regresión era capaz de calcular los bounding boxes. Sin embargo, con el impacto de las Redes Convolucionales, estas técnicas pasaron a un segundo plano, convirtiéndose las CNNs en el “estado del arte” de la detección de objetos. Este avance ha permitido aplicarlo a tareas como la conducción autónoma, la detección de anomalías o la videovigilancia.

En los modelos dedicados a la detección de objetos de manera supervisada se pueden distinguir dos partes: el encoder y el decoder.

1. Encoder: coge la imagen y la pasa a través de una serie de capas y bloques para extraer características útiles para la localización y clasificación.
2. Decoder: se dedica a predecir los bounding boxes y las etiquetas.

El convenio que existe para medir la bondad de la solución dada es usar la métrica **Intersection Over Union (IoU)**: dados dos bounding boxes (el predicho por el modelo y el etiquetado real), se calcula el área de intersección y se divide por el área de la unión. Por lo tanto, este IoU toma valores en $[0, 1]$ donde el 0 es un muy mal desempeño (no hay intersección) y un 1 es muy buen desempeño (ambos bounding boxes coinciden). [23]

$$IoU = \frac{\text{Área de la intersección}}{\text{Área de la unión}} \quad (9)$$

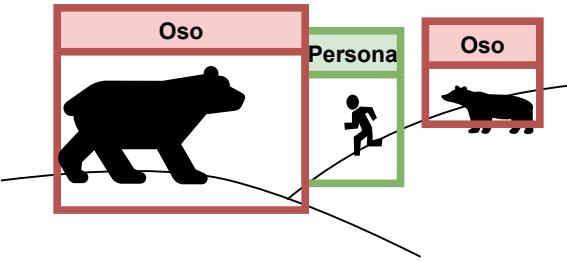


Figura 11: Ejemplo esquemático de cómo sería resuelta una tarea de detección de objetos con bounding boxes.

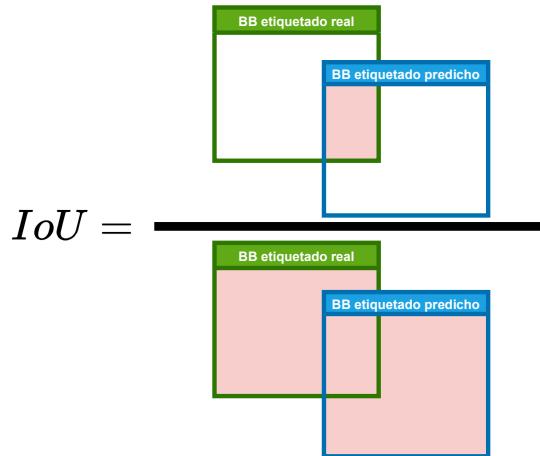


Figura 12: Explicación gráfica de **Intersection Over Union (IoU)** (9) con bounding boxes

3

XAI EXPLAINABLE AI

3.1 XAI

Actualmente la Inteligencia Artificial pasa por un momento de gran popularidad y uso en una gran variedad de disciplinas. Esta popularidad es debida a los importantes avances hechos en la última década en este campo, avances que han permitido abordar problemas de alta complejidad con un rendimiento terriblemente bueno. Los modelos capaces de abordar estas tareas no requieren prácticamente de intervención humana para diseñarse o tomar decisiones. Es por ello que cuando la tarea o problema a resolver afecta directamente a las personas surge una necesidad de entender al modelo, de ser capaz de preguntarle: ¿por qué?, ¿cómo?... y de que el modelo sea capaz de dar una explicación.

A pesar de que los primeros modelos de Inteligencia Artificial que surgieron eran muy interpretables, conforme pasan los años esta interpretabilidad se va perdiendo, surgiendo cada vez modelos de decisión más opacos (v.g. Redes Neuronales Profundas - DNNs). Siguiendo con este ejemplo, las DNNs tienen ese éxito debido a la eficiencia de los algoritmos de aprendizaje usados y a su enorme espacio de parámetros. La gran magnitud que conlleva el número de parámetros del modelo hace considerar a estos como **modelos de caja negra**. Tenemos así, una relación inversa entre dos conceptos: **modelos transparentes** y **modelos de caja negra**. Con el paso del tiempo el uso de estos modelos de caja negra es cada vez más frecuentes en áreas como la conducción automática de coches, finanzas, seguridad o medicina en donde la transparencia de estos modelos para su entendimiento es algo crucial.

Aunque en primera instancia se pueda pensar que aumentar la transparencia implique reducir el rendimiento general de los modelos, no es así. La transparencia ayudará a entender qué está pasando y por qué se toman ciertas decisiones; detectando sesgos, perturbaciones pequeñas que afecten en gran magnitud a la decisión o las variables más importantes del problema.

Desde esta perspectiva vemos como el área de la Inteligencia Artificial se encuentra actualmente ante una barrera, la explicabilidad. Así, surge la **IA explicable (eXplai-**

nable AI - XAI) Arrieta et al. [1] con el objetivo de crear nuevas técnicas de Machine Learning de forma que:

1. Produzca modelos más explicables manteniendo el rendimiento
2. Permita a los humanos entender estos modelos incrementando tanto la confianza como el manejo efectivo sobre ellos.

En este trabajo nos vamos a centrar en un problema relacionado con la explicabilidad en Redes Convolucionales. Estas redes constituyen actualmente el “*estado del arte*” en muchas de las tareas de Visión por Computador. Las técnicas existentes que tienen como objetivo entender estas redes siguen dos enfoques principales:

1. Entender el proceso de decisión mapeando hacia atrás para ver qué áreas del input son relevantes para el output.
2. Accediendo al interior de la red, interpretar cómo las capas intermedias ven el mundo exterior. Esto quiere decir, que no se fijan únicamente en un input específico, sino en general.

En concreto, el trabajo se centrará en técnicas relativas al primer punto. Gracias a las capas de activación de la red y su paso por ella, se puede trazar una relación entre las regiones de la entrada y el valor de la salida, permitiendo localizar qué áreas han influido más, dando lugar a un problema de **detección de objetos débilmente supervisada**. La técnica denominada *Class Activation Maps (CAM)*, propone una modificación de la red original que nos permite obtener mapas de calor sobre la entrada para identificar las regiones relevantes, localizando así, presencia de una clase. Esta técnica da lugar a otras que serán también objeto de estudio: *Gradient-weighted Class Activation Mapping (Grad-CAM)*, *Grad-CAM++* y *Smooth Grad-CAM++*

3.2 CLASS ACTIVATION MAP

En Zhou et al. [2] se presenta la técnica *Class Activation Map* o *CAM*. Es una técnica usada para generar mapas de activación sobre el input de la red, indicando las regiones usadas por la CNN para identificar cada clase. Estos mapas de activación otorgan **explicabilidad** al modelo, permitiendo entender en qué se ha fijado la red para realizar una determinada predicción.

3.2.1 Idea

La investigación realizada por los mismos autores en Zhou et al. [24] fue lo que dio pie a esta técnica. En Zhou et al. [24] muestran cómo redes convolucionales entrenadas de manera supervisada para clasificación, tienen un enorme potencial para localizar objetos de manera no supervisada. Sin embargo, este conocimiento se pierde en el paso por las capas totalmente conectadas.

Como primera aproximación para solventar el problema se presentó el artículo Bazzani et al. [25] donde proponen modificar la red añadiendo un *Global Max Pooling* (GMP) seguido de una capa lineal justo después de la última capa convolucional, usándose como clasificador en el lugar de las capas totalmente conectadas. Rápidamente se consiguió mejorar la técnica en una nueva publicación Zhou et al. [2], que contempla la técnica que tenemos como objeto en esta sección. Esta técnica propone sustituir la capa GMP por una *Global Average Pooling* (GAP).

En el artículo se muestra cómo el desempeño de la red mejora con este cambio. Esto es porque al aplicar una media global, se están teniendo en cuenta todos los valores presentes en las capas de activación del extracto de características. Sin embargo, en el GMP únicamente se tiene en cuenta un valor, esto implica que la red va a intentar minimizar la pérdida basándose solo en la región que discrimina ese valor e incluso puede meter ruido en el resultado final. Este problema se verá solucionado con CAM (Zhou et al. [2]) gracias al uso de un *Global Average Pooling* (GAP). El GAP permite optimizar el modelo maximizando todas las regiones relevantes de la imagen (y no solo una), creando mapas de activación de clase más completos y con más información. Esta capa GAP irá seguida de una capa lineal que servirá de clasificador. De esta manera, se propone un cambio en la red original con el fin de conservar la capacidad de localización de esta red.

3.2.2 Método

En esta sección se va a describir la técnica CAM en CNNs: consiste en, dada una CNN, coger sus capas convolucionales y poner justo después de ellas:

1. Un *Global Average Pooling* (GAP)
2. Una *Fully Connected layer* (FC)

Estas nuevas quedarán justo antes de la capa final (v.g. Softmax para clasificación). Para conseguir el *Class Activation Map* basta con usar la información dada por los pesos de la última capa de activación de la red, ponderando cada mapa con los pesos de la capa FC añadida.

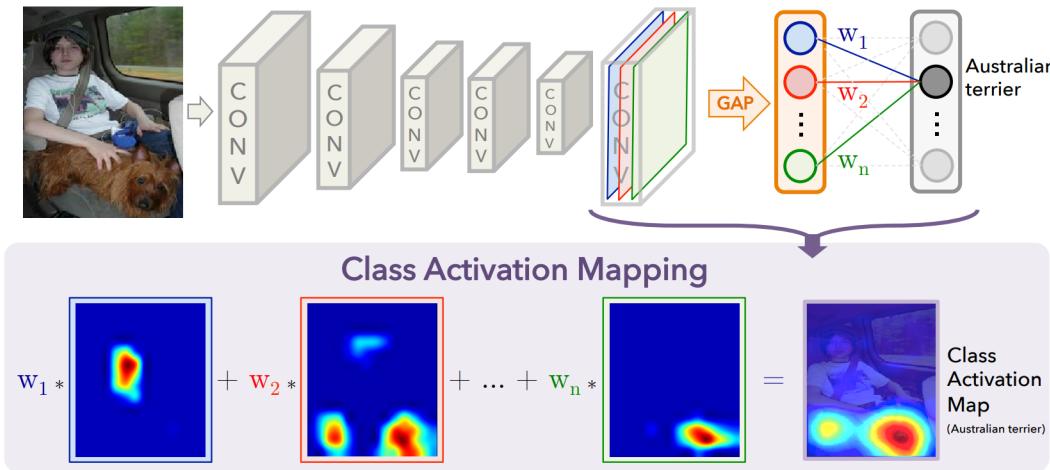


Figura 13: Los mapas de activación son reescalados al tamaño de la imagen y la suma ponderada dada por la capa FC de ellos nos da el CAM , que resalta las regiones discriminatorias de la clase en la imagen. (Fig 2. - Zhou et al. [2])

Siendo A^k al mapa de activación k -ésimo de la salida de la última capa convolucional y w_k^C al k -ésimo peso de la capa totalmente conectada FC para la clase C , tenemos que la salida de la red es:

$$\begin{aligned} Y^C &= \sum_k w_k^C \underbrace{\left(\frac{1}{Z} \sum_i \sum_j A_{i,j}^k \right)}_{Global\ Average\ Pooling} \\ &= \sum_k \frac{1}{Z} \sum_i \sum_j (A_{i,j}^k w_k^C) \end{aligned} \quad (10)$$

Donde $Z = alto(A) \times ancho(A)$, siendo A la última capa de activación de la red. En la [ecuación 10](#), podemos aprovechar lo que tenemos entre paréntesis, definiendo $F_k^C = A^k w_k^C$. Este nuevo mapa F_k^C nos da una activación parcial de la clase C en cada k . El **Mapa de Activación de la Clase (CAM)**, L_{CAM}^C ([ecuación 11](#)), se calculará como la suma de todas ellas.

$$L_{CAM}^C = \sum_k F_k^C = \sum_k A^k w_k^C \quad (11)$$

Este procedimiento se puede ver reflejado gráficamente en la [Figura 13](#), donde las imágenes de activación representarían los $A^k, k \in \{1, 2, \dots, n\}$, escalados al tamaño de la imagen.

3.3 GRADIENT-WEIGHTED CLASS ACTIVATION MAPPING

La técnica de *CAM* se restringe a un tipo de redes, dejando fuera a muchas otras, por la necesidad de no tener varias capas totalmente conectadas *FC* en la región clasificadora de la red. Con la idea de solventar esta restricción o hacerla más débil surge **Gradient-weighted Class Activation Mapping (Grad-CAM)** (Selvaraju et al. [3]).

3.3.1 Método

La novedad de *Grad-CAM* viene en el cálculo de los pesos w_k^C que ponderan los mapas de activación A^k ([ecuación 11](#)). Se propone calcular el gradiente de la salida con respecto el mapa de activación de la última capa convolucional (en el paper, Selvaraju et al. [3], comentan que se puede aplicar a cualquier capa, nosotros nos centraremos en la salida), esto es $\frac{\partial y^C}{\partial A^k}$. Aplicándole un *GAP* obtendríamos el peso, que ahora llamaremos α_k^C ([ecuación 12](#)) para evitar confusiones con los pesos anteriores.

$$\alpha_k^C = \overbrace{\frac{1}{Z} \sum_i \sum_j}^{GAP} \frac{\partial Y^C}{\partial A_{i,j}^k} \quad (12)$$

El uso de los gradientes pretende dar un sentido a su capa de activación: el gradiente será mayor cuanto más aporte este mapa a la salida. Esta importancia se verá ahora reflejada en el mapa de activación de clase. En la [ecuación 13](#) está el cálculo propuesto en el paper Selvaraju et al. [3] de este mapa de activación de clase. Argumentan usar una función *ReLU* para destacar solamente las zonas que aportan de manera positiva a la clase, ya que metiendo zonas de aporte negativo se destacarían zonas que pertenecen a otras clases.

$$L_{Grad-CAM}^C = ReLU \left(\sum_k A^k \alpha_k^C \right) \quad (13)$$

3.3.2 Generalización de CAM

La realidad de este método es que generaliza al anterior. Supongamos que tenemos ahora una red válida para CAM: con un *GAP* seguido de una capa *FC* justo después de la salida de la última capa convolucional. En esta situación teníamos que:

$$\begin{aligned}
 Y^C &= \sum_k w_k^C \underbrace{\frac{1}{Z} \sum_i \sum_j A_{i,j}^k}_{G^k} \\
 &= \sum_k w_k^C G^k
 \end{aligned} \tag{14}$$

Donde se ha definido $G^k = \frac{1}{Z} \sum_i \sum_j A_{i,j}^k$. Calculamos ahora el gradiente de la salida de la clase C con respecto a G^k :

$$\frac{\partial Y^C}{\partial G^K} = \frac{\frac{\partial Y^C}{\partial A_{i,j}^k}}{\frac{\partial G^K}{\partial A_{i,j}^k}}$$

De la [ecuación 14](#) sacamos que $\frac{\partial G^k}{\partial A_{i,j}^k} = \frac{1}{Z}$ y que $\frac{\partial Y^C}{\partial G^k} = w_k^C$, con lo que

$$\begin{aligned}
 w_k^C &= Z \frac{\partial Y^C}{\partial A_{i,j}^k} \\
 &\Downarrow \\
 \sum_i \sum_j w_k^C &= \sum_i \sum_j Z \frac{\partial Y^C}{\partial A_{i,j}^k} \\
 &\Updownarrow \\
 Z w_k^C &= Z \sum_i \sum_j \frac{\partial Y^C}{\partial A_{i,j}^k} \\
 &\Updownarrow \\
 w_k^C &= \sum_i \sum_j \frac{\partial Y^C}{\partial A_{i,j}^k} \\
 &\Updownarrow \\
 w_k^C &= Z \alpha_k^C
 \end{aligned} \tag{15}$$

Concluyendo con la [ecuación 15](#), Grad-CAM aporta unos pesos idénticos a CAM salvo una constante de proporcionalidad que no influye en el mapa de activación de clase final. Así, se demuestra que **Grad-CAM es una generalización de CAM**, permitiendo ahora aplicarse a muchas más CNNs.

3.4 GRAD-CAM++

Con una buena generalización de [CAM](#), se ha conseguido un método que generaliza y permite su uso en una gran variedad de redes: [Grad-CAM](#). Ahora, se busca mejorar la técnica con una ponderación, ya no de los mapas de activación de la salida de la última capa convolucional, sino una ponderación a nivel de píxel de los pesos $\alpha_k^C, k \in \{1, 2, \dots, n\}$ calculados en [Grad-CAM](#) ([ecuación 12](#)). El objetivo de esta nueva técnica, **Grad-CAM++** (Chattpadhyay et al. [4]), no es recoger únicamente la importancia del mapa A^k , sino la importancia relativa del píxel al mapa que lo activa.

Este cambio pretende mejorar la visualización para aquellas imágenes en las que haya más de una instancia de la clase o características distintas de una clase situadas en distintas partes de la misma, dándoles una misma relevancia en el *mapa de activación de clase*.

3.4.1 Método

Como hemos mencionado, [Grad-CAM++](#) intenta hacer una mejora ponderando los pesos $\alpha_k^C, k \in \{1, 2, \dots, n\}$ calculados en [Grad-CAM](#) ([ecuación 12](#)). Los pesos de esta nueva ponderación vendrán marcados bajo la letra β . En concreto, en esta técnica los pesos β_k^C tienen la forma descrita en la [ecuación 16](#).

$$\beta_k^C = \sum_i \sum_j \gamma_{i,j}^{C,k} \text{ReLU} \left(\frac{\partial Y^C}{\partial A_{i,j}^k} \right) \quad (16)$$

Nos fijamos que si $\gamma_{i,j}^{C,k} = \frac{1}{Z} \forall i, j \implies \beta_k^C = \alpha_k^C$. Para calcular estos pesos $\gamma_{i,j}^{C,k}$, en Chattpadhyay et al. [4] parten de la [ecuación 17](#):

$$Y^C = \sum_k \left(\underbrace{\left[\sum_a \sum_b \gamma_{a,b}^{C,k} \text{ReLU} \left(\frac{\partial Y^C}{\partial A_{a,b}^k} \right) \right]}_{\beta_k^C} \left[\sum_i \sum_j A_{i,j}^k \right] \right) \quad (17)$$

Destacar que el cálculo de Y^C no está sujeta a la combinación lineal de la [ecuación 17](#), esta igualdad se fuerza simplemente para el cálculo de los pesos $\gamma_{i,j}^{C,k}$. Para el cálculo de estos pesos hay que calcular las derivadas parciales con respecto a $A_{i,j}^k$ a ambos lados de [ecuación 17](#).

$$\frac{\partial Y^C}{\partial A_{i,j}^k} = \sum_a \sum_b \gamma_{a,b}^{C,k} \frac{\partial Y^C}{\partial A_{a,b}^k} + \sum_a \sum_b A_{a,b}^k \left[\gamma_{i,j}^{C,k} \frac{\partial^2 Y^C}{(\partial A_{i,j}^k)^2} \right]$$

$$\frac{\partial^2 Y^C}{(\partial A_{i,j}^k)^2} = 2\gamma_{a,b}^{C,k} \frac{\partial^2 Y^C}{(\partial A_{a,b}^k)^2} + \sum_a \sum_b A_{a,b}^k \left[\gamma_{i,j}^{C,k} \frac{\partial^3 Y^C}{(\partial A_{i,j}^k)^3} \right]$$

Reordenando términos:

$$\gamma_{i,j}^{C,k} = \frac{\frac{\partial^2 Y^C}{(\partial A_{i,j}^k)^2}}{2\frac{\partial^2 Y^C}{(\partial A_{i,j}^k)^2} + \sum_a \sum_b A_{a,b}^k \left[\frac{\partial^3 Y^C}{(\partial A_{i,j}^k)^3} \right]}$$

En este razonamiento solo le hemos pedido a la predicción de clase Y^C que sea lo suficientemente suave para poder calcular, como mínimo, sus derivadas parciales de orden 3.

El mapa de activación por lo tanto sería:

$$L_{Grad-CAM++}^C = \text{ReLU} \left(\sum_k A^k \beta_k^C \right) \quad (18)$$

3.4.2 Análisis computacional

Para esta técnica hay que pararse para hablar de la implementación. Para calcular las derivadas parciales se requiere de mucha memoria para guardar los grafos de PyTorch. En el paper de Chattopadhyay et al. [4] proponen una solución para este problema.

Si S^C es la salida de nuestra red justo antes del *softmax*, definimos:

$$Y^C = \exp(S^C)$$

$$\frac{\partial Y^C}{\partial A_{i,j}^k} = \exp(S^C) \frac{\partial S^C}{\partial A_{i,j}^k} \quad (19)$$

El paso de S^C por una exponencial permite que el cálculo de derivadas sea fácil. Como la exponencial es una función infinitamente derivable, no perdemos propiedades y podemos seguir aplicando nuestro método de *Grad-CAM++*.

$$\frac{\partial^2 Y^C}{(\partial A_{i,j}^k)^2} = \exp(S^C) \left[\left(\frac{\partial S^C}{\partial A_{i,j}^k} \right)^2 + \frac{\partial^2 S^C}{(\partial A_{i,j}^k)^2} \right] \quad (20)$$

- NOTA: Si $f(x) = \max(0, x)$ es la función ReLU definida en \mathbb{R} , tenemos que

$$\begin{aligned} \frac{\partial f}{\partial x} &= \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases} \\ \frac{\partial^2 f}{\partial x^2} &= 0 \quad \forall x \in \mathbb{R} \end{aligned} \quad (21)$$

S^C , al ser resultado de composición de funciones lineales y ReLUs, usando la regla de la cadena y la [ecuación 21](#), se puede concluir que

$$\frac{\partial^2 S^C}{(\partial A_{i,j}^k)^2} = 0 \quad (22)$$

Sustituyendo la [ecuación 22](#) en la [ecuación 20](#):

$$\frac{\partial^2 Y^C}{(\partial A_{i,j}^k)^2} = \exp(S^C) \left(\frac{\partial S^C}{\partial A_{i,j}^k} \right)^2 \quad (23)$$

Razonando de la misma manera obtenemos

$$\frac{\partial^3 Y^C}{(\partial A_{i,j}^k)^3} = \exp(S^C) \left(\frac{\partial S^C}{\partial A_{i,j}^k} \right)^3 \quad (24)$$

Luego los pesos que ponderan β_k^C quedarían:

$$\gamma_{i,j}^{C,k} = \frac{\left(\frac{\partial S^C}{\partial A_{i,j}^k} \right)^2}{2 \left(\frac{\partial S^C}{\partial A_{i,j}^k} \right)^2 + \sum_a \sum_b A_{a,b}^k \left(\frac{\partial S^C}{\partial A_{i,j}^k} \right)^3} \quad (25)$$

De esta manera habríamos conseguido nuestro objetivo de que en un solo paso por el grafo tendríamos el cálculo de los pesos hecho.

3.5 SMOOTH GRAD-CAM++

Hasta ahora se ha seguido un proceso de generalización de técnicas para visualizar *mapas de activación de clase*. La más general que se ha visto es [Grad-CAM++](#). En este

punto, toca ver una nueva técnica que generaliza desde otro punto de vista: ya no se busca darle más sentido a los mapas de activación de características cambiando pesos, ahora el objetivo de mejora se encuentra en la entrada de la red. La idea es que el ruido en la imagen puede afectar a la decisión, así que lo que buscamos con esta técnica, llamada **Smooth Grad-CAM++** (Omeiza et al. [5]) es eliminar el ruido de la imagen para tener una interpretabilidad limpia de la red a ojos de un humano. Para ello, se hace uso de una técnica: **Smooth Grad** (Smilkov et al. [26])

3.5.1 Smooth Grad

Es una técnica usada para suavizar la imagen. Esta técnica está presentada en Smilkov et al. [26], y su ingeniosa forma de *quitar ruido añadiendo ruido* será de gran utilidad para el cometido final de la sección: los *mapas de activación de clase*.

Considerando un problema de clasificación de imágenes en un conjunto de clases. Dada una imagen de input x tenemos nuestra salida S^C para cada clase C . Si la función S^C es diferenciable, para cada imagen x podemos definir un *sensitivity map*:

$$M_C(x) = \frac{\partial S^C(x)}{\partial x} \quad (26)$$

Intuitivamente, M_c representa cuánto cambia la clasificación de la red para un x input con pequeños cambios en x . El ruido de los *sensitivity maps* se debe a que a pequeñas escalas, S^C puede variar enormemente. No hay porqué pensar que estas derivadas varíen de forma suave, parte de la culpa la tienen las activations ReLU en las que están basadas las redes.

Como solución a este ruido proponen el cálculo de ∂S^C con un kernel Gausiano. Este kernel es calculado como aproximación metiendo ruido a la imagen con n muestras aleatorias de ruido Gausiano \mathcal{N} , dándole de media 0, desviación σ y sumándole el input x .

$$\hat{M}_c(x) = \frac{1}{n} \sum_1^n M_c(x + \mathcal{N}(0, \sigma^2)) \quad (27)$$

Este método será denominado **Smooth Grad**.

3.5.2 Método

La técnica para generar *mapas de activación de clase* usando *Smooth Grad* se llama **Smooth Grad-CAM++** (Omeiza et al. [5]). La idea consiste en generar nuevas imágenes de input y usar la técnica de *Grad-CAM++* de manera conjunta.

El primer paso, es generar las imágenes con ruido con *Smooth Grad*. Se generan n imágenes partiendo la imagen original con desviación σ :

$$\begin{aligned}x_0 &= x \\x_1 &= x + \mathcal{N}(0, \sigma^2) \\x_2 &= x + \mathcal{N}(0, \sigma^2) \\&\dots \\x_n &= x + \mathcal{N}(0, \sigma^2)\end{aligned}$$

Tras pasarlas por la red, recogemos las derivadas parciales de cada uno de los x_t , $t \in \{0, 1, \dots, n\}$. Así notaremos como $D_p^{k,t}$ a la p -ésima derivada parcial de x_t con respecto al mapa de características k . Así, los pesos γ calculados en *Grad-CAM++* pasarían a ser:

$$\Gamma_{i,j}^{C,k} = \frac{\frac{1}{n+1} \sum_{t=1}^{n+1} D_{2,i,j}^{k,t}}{2 \frac{1}{n+1} \sum_{t=1}^{n+1} D_{2,i,j}^{k,t} + \sum_a \sum_b A_{a,b}^k \left[\frac{1}{n+1} \sum_{t=1}^{n+1} D_{3,i,j}^{k,t} \right]} \quad (28)$$

En el caso $n = 0$, $\Gamma_{i,j}^{C,k} = \gamma_{i,j}^{C,k}$. Ahora definimos los nuevos pesos que acompañaran a los mapas de activación de características:

$$\omega_k^C = \sum_i \sum_j \Gamma_{i,j}^{C,k} \text{ReLU} \left(\frac{1}{n+1} \sum_{t=1}^{n+1} D_{1,i,j}^{k,t} \right) \quad (29)$$

Con estos pesos ya se podría generar el mapa de activación de clase

$$L_{SmoothGrad-CAM++}^C = \text{ReLU} \left(\sum_k A^k \omega_k^C \right) \quad (30)$$

Parte III

APLICACIÓN

En esta parte del trabajo se verá la aplicación de la teoría vista hasta ahora a un problema de diagnóstico en imágenes médicas.

4

APLICACIÓN DE LAS TÉCNICAS DE EXPLICABILIDAD AL DIAGNÓSTICO DE CÁNCER EN IMÁGENES HISTOLÓGICAS DE PRÓSTATA

En esta parte se van a aplicar las técnicas de explicabilidad vistas para conseguir mapas de calor sobre la entrada de una red convolucional a imágenes médicas. En concreto, se van a aplicar a imágenes histológicas de próstata obtenidas a partir de biopsias para la detección de tejido cancerígeno.

Se va hacer un estudio con 4 redes distintas: VGG16 [11], RESNET [12], MOBILE-NET [13] y EFFICIENTNET B0 [14]; y con las 4 técnicas de explicabilidad vistas: **CAM** [2], **Grad-CAM** [3], **Grad-CAM++** [4] y **Smooth Grad-CAM++** [5]. Se pretende hacer una comparación entre modelos y una comparación entre técnicas. De esta manera, con la comparación entre modelos, se podrán observar dependencias en la bondad de la localización con respecto a la capacidad de clasificación del modelo, por ejemplo. Por otro lado, se compararán las técnicas entre sí viendo qué resultados da cada una y sacar conclusiones sobre qué técnica sería la mejor para este caso de uso. Así se podrán obtener conclusiones con el mejor modelo con la mejor técnica sobre si son precisos a la hora de localizar regiones con tejido cancerígeno.

4.1 DATASET

El conjunto de datos sobre el que se van a aplicar las técnicas de explicabilidad que se han visto en el capítulo anterior es de imágenes histológicas de próstata obtenidas a partir de biopsias. El dataset es **SICAPv1** [27] y está compuesta por 79 imágenes histológicas de próstata, llamadas WSI (Whole Slide Image: capturan imágenes de secciones de tejido mosaico a mosaico o de forma lineal. Los mosaicos se capturan y se ensamblan digitalmente para generar una imagen digital completa). Las imágenes han sido recogidas por especialistas del Hospital Clínico Universitario de Valencia.

El dataset procesado en [27] está compuesto por 79 WSI: 19 corresponden a biopsias de tejido benigno de próstata y 60 corresponden a biopsias de tejido patológico.

Las WSI han sido divididas en tamaños de 512×512 con un 50 % de superposición ([Imagen a]) de la Figura 15). Dando lugar al reparto del Cuadro 1.

	Benigno	Patológico
Train	17	43
Test	2	17

Cuadro 1: División de las WSI del dataset original .

4.1.1 Procesado del dataset

Como bien se dijo en la introducción del problema, se van a usar las redes: VGG, RESNET, MOBILENET y EFFICIENTNET. Todas estas redes aceptan como entrada imágenes con dimensiones 224×224 , por lo que se va a procesar el dataset para obtener imágenes de estas dimensiones.

Cada imagen de 512×512 se va a dividir en imágenes de 224×224 uniformemente distribuidas sobre la original (solapándose el mismo porcentaje dos a dos contiguas). Esto generará 9 imágenes nuevas con solapamiento. Como las imágenes de 512×512 ya tenían un solapamiento del 50 % entre imágenes contiguas, esto va a provocar tener imágenes muy redundantes en el conjunto de datos. La solución es, por tanto, eliminar la superposición del 50 % que trae el dataset de inicio. Para ello hacemos dos selecciones disjuntas de las imágenes de 512:

- *Selección par*: imágenes cuya fila y columna del WSI (imagen original) sea par.¹
- *Selección impar*: imágenes cuya fila y columna del WSI sea impar.

Para verlo gráficamente, en la Figura 15 se ve cómo se hace esta selección en las Imágenes b) y c). Por último, se procesa cada una de las imágenes de cada selección dividiéndola en 9 imágenes de 224×224 , Imagen d) de la Figura 15 cogiendo únicamente aquellas que tengan más de un 25 % de píxeles etiquetados con la clase a la que pertenecen. Quedando al final la distribución del Cuadro 2. En lo que sigue se tendrá en cuenta la **selección par**.

¹ La fila y columna hacen referencia al desplazamiento de la imagen con respecto a la esquina superior izquierda.

	PAR		IMPAR	
	Benigno	Patológico	Benigno	Patológico
Train	9110	2344	9033	2325
Validation	2277	585	2258	581
Test	4014	3322	3751	3233

Cuadro 2: Distribución de las imágenes del dataset final tras hacer las selecciones par e impar.

Balanceo

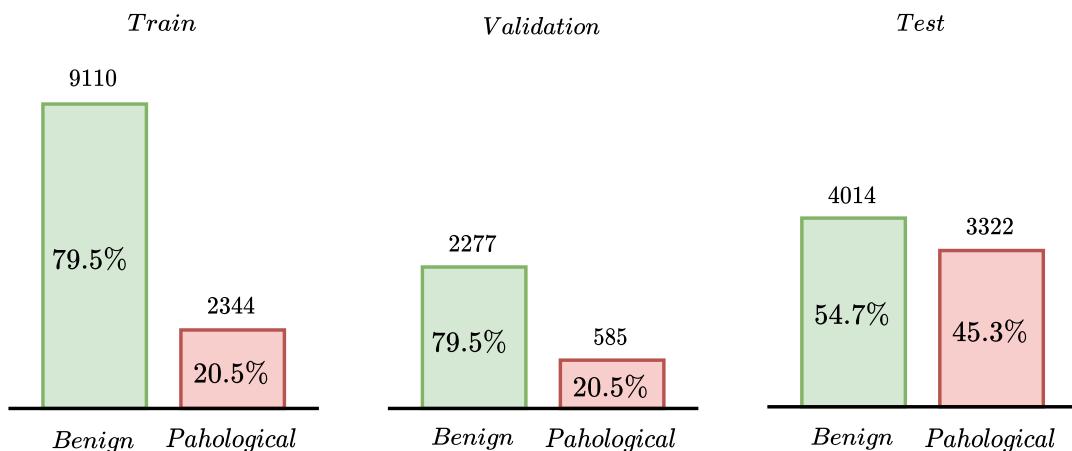


Figura 14: Distribución de datos en la selección par.

Con ayuda de las gráficas de la Figura 14 se puede observar el desbalanceo de clases que hay dentro de los datos de validación y train. Esto puede deberse a que es una proporción de muestras benignas y patológicas que generalizan los casos reales. Sin embargo, este desbalanceo también puede ser fuente de muchos falsos negativos (casos patológicos predichos como benignos) a la hora de probar el modelo fuera de los datos. De cualquier manera, en los datos de test sí que tenemos ambas clases bien balanceadas. Como el interés de este problema subyace en los casos patológicos, este aumento de proporción patológica va a servir para aumentar la confianza en los resultados que se tienen con el modelo al predecir esta clase.

En cierta manera, si se consiguen buenas métricas en test, se tendrá la seguridad de que el modelo es capaz de conseguir buenos resultados con pocos datos patológicos. Esto es algo importante, ya que en la población humana, la proporción de personas con patología de este tipo es minoritaria. A esto hay que sumarle que la muestra de biopsia de una persona patológica contenga tejido patológico, lo que sigue dificultando el acceso a datos de tejido patológico.

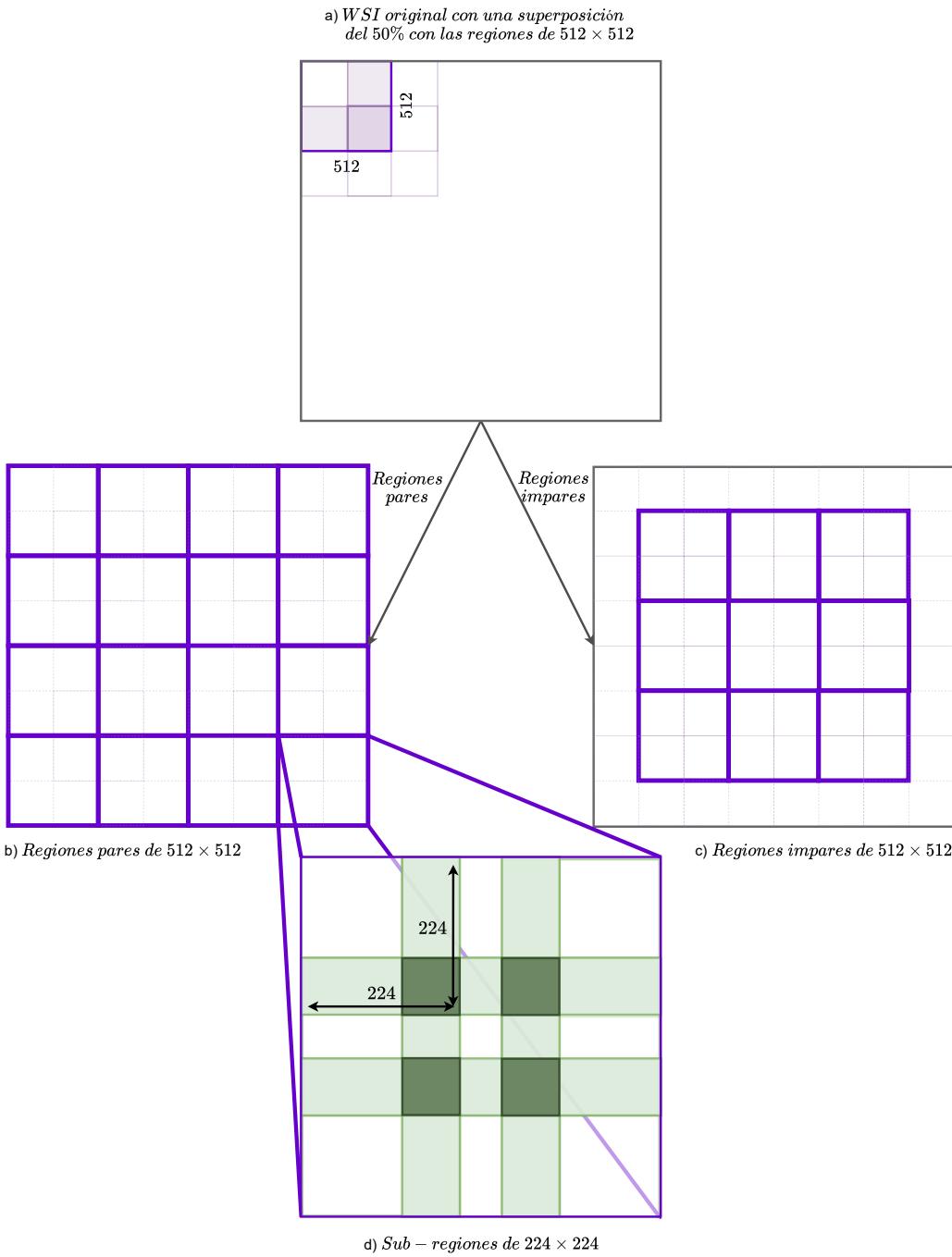


Figura 15: Procesado dataset original al dataset final. En la primera imagen a) de arriba se muestra una imagen WSI completa, donde se han cogido regiones de tamaño 512×512 con una superposición del 50 %. Lo que se busca en el dataset final es tener imágenes de tamaño 224×224 . Para evitar redundancia, se elimina la superposición del 50 % original. Para ello, hay que eliminar la superposición metida al inicio cogiendo las imágenes pares (aquellas que están en una fila y columna par) Imagen b) o las imágenes impares (aquellas que están en una fila y en una columna impar) Imagen c). Cogiendo bien la selección par, bien la selección impar, se divide cada imagen de la selección en 9 regiones de 224×224 , como en la Imagen d), habrá una superposición de 80 píxeles dentro de cada imagen de la selección, señalada esta superposición en verde en la Imagen d)

Máscaras

En el conjunto de datos existen varios tipos de máscaras en función de la información que representan:

- Tejido: recogen toda la región donde hay tejido.
- Patológica: recogen únicamente el tejido patológico.

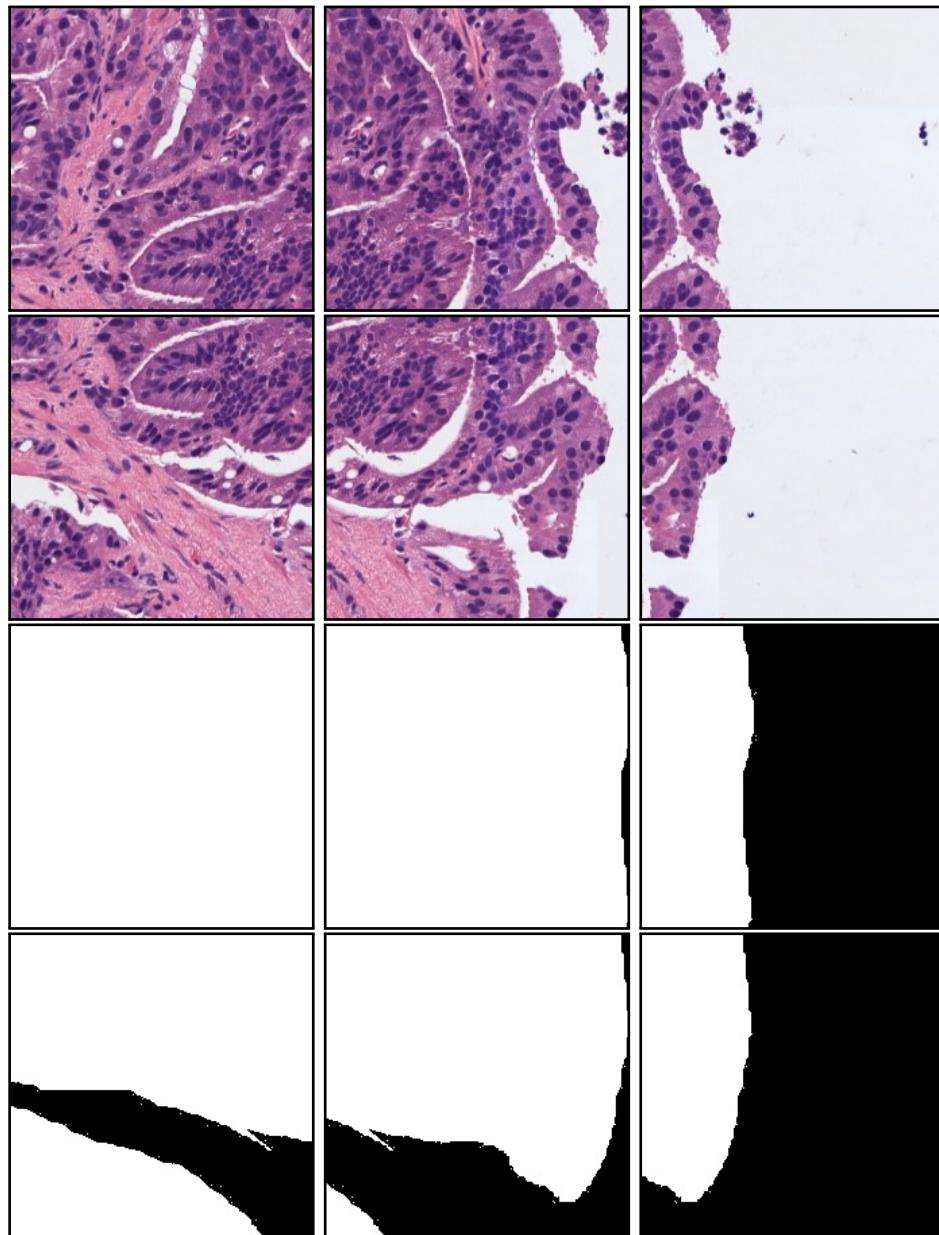


Figura 16: Muestra de ejemplo cancerígeno junto a sus máscaras cogida de los datos procesados de entrenamiento. Se puede observar el solapamiento entre las imágenes.

Un aspecto importante detectado al estudiar el etiquetado de las máscaras es que estas no son 100 % precisas. por ejemplo en las imágenes de la derecha de la [Figura 16](#) se ven como las máscaras recogen fondo blanco como tejido cancerígeno, provocando de inicio un error intrínseco. Esto tendrá repercusión a la hora de evaluar la bondad de localización de nuestros modelos y técnicas.

4.2 REDES UTILIZADAS

Como bien se ha repetido con anterioridad, las redes base que se van a usar para estudiar las distintas técnicas de explicabilidad van a ser VGG16 [11], RESNET [12], MOBILENET [13] y EFFICIENTNET B0 [14].

	Input size	Tamaño (Mb)	Top-1 Accuracy	Top-5 Accuracy	Parámetros (Millones)	Profundidad
VGG 16	224×224	512	71.3 %	90.1 %	138.4	16
ResNet 18	224×224	43	69.3 %	88.94 %	11	18
MobileNet v2	224×224	14	71.3 %	90.1 %	3.5	105
EfficientNet b0	224×224	29	77.1 %	93.3 %	5.3	132

Cuadro 3: Estadísticas de las redes usadas sobre el conjunto de datos de ImageNet [28], conjunto de datos con el que se han preentrenado las redes. Datos de Keras [29] y del estudio de Gupta et al. [30].

La selección de estas redes se ha escogido con la intención de representar arquitecturas que tuvieron un impacto importante dentro del mundo del Deep Learning por la introducción de algunos conceptos o técnicas. **VGG16** [11] es una red clásica (la más antigua de las cuatro) cuya simpleza a nivel de capas, su baja profundidad y su capacidad de desempeño hace que sea una red que, a pesar de su tiempo, sigue estando muy presente en todos los estudios.

A partir de redes como VGG16 se empiezan a introducir mejoras en base a los problemas que iban surgiendo, como por ejemplo el aumento de la profundidad de la red. Poniendo en contexto, una red es un *aproximador universal*² y se vio cómo aumentando el número de capas (la profundidad), se podían reducir el tamaño de las capas de manera que se obtuviesen mismos resultados. Lo interesante era que con el aumento de profundidad se podían reducir el número de parámetros totales en la red (surge entonces el interés en el Deep Learning). Pero la profundidad daba un problema conocido como *desvanecimiento de gradiente*³. Surge entonces **RESNET** [12], que introdujo el concepto de *bloques residuales* ([Figura 17](#)) para solucionar el problema. En concreto, en este trabajo se va a utilizar la versión *RESNET-18*.

La introducción de las aplicaciones de las redes profundas en los dispositivos que utiliza la gente de manera usual provocó situar el foco de mejora en el espacio de

² Una red puede llegar a aproximar cualquier función tanto como se quiera modificando su arquitectura.

³ Problema durante backpropagation con el gradiente donde los valores del gradiente se desvanecen conforme llegan a la capa de entrada. Esto provoca que los pesos no se actualicen como deben.

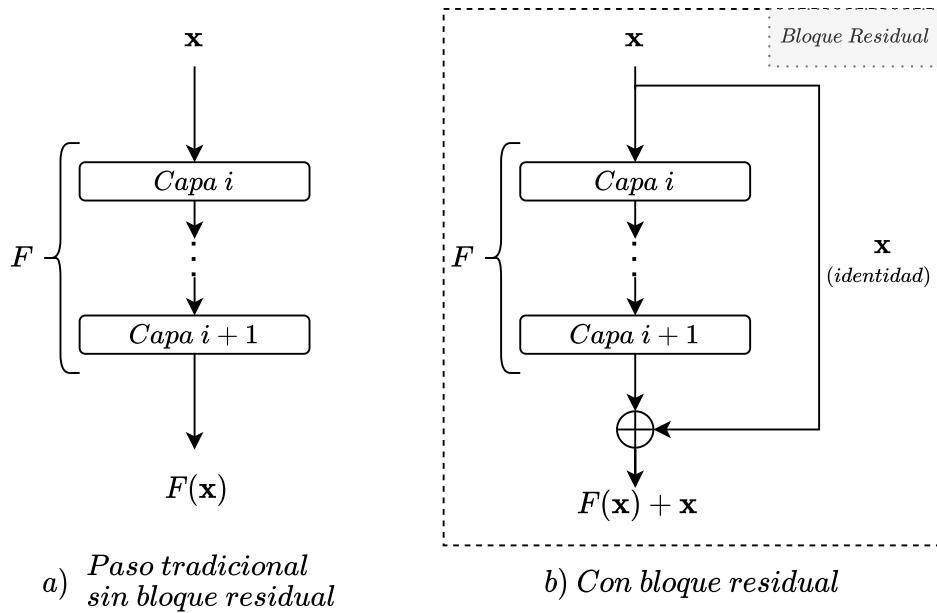


Figura 17: Esquema gráfico de bloque residual. Se pasa el valor de entrada al bloque x al final del bloque sumándolo al valor que se ha obtenido al pasar ese input por el bloque *feedforward* F .

memoria que necesitaban las redes, **MOBILENET** [13] consiguió un tipo de red que conseguía estos propósitos, en el [Cuadro 3](#) se puede ver cómo su tamaño en Mb es muy pequeño y esto pudo permitir incorporar los avances de estas redes en dispositivos como móviles de uso cotidiano. En este trabajo se usará la versión *MOBILENET V2*.

El avance dado por **EFFICIENTNET** [14] tiene que ver en la forma de buscar una red con la mejor compensación *Accuracy - Número de Parámetros*. En Tan et al. [14] proponen un método para hacerlo, consiguiendo muchas redes distintas apodadas bajo el distintivo de bX , variando entre ellas el número de parámetros. En este trabajo se va a usar la versión más pequeña: *EFFICIENTNET B0*.

4.2.1 Implementación

Las 4 redes a usar ([Cuadro 3](#)) están preentrenadas con ImageNet [28]. Para adaptarlas al problema de clasificación binaria al que se las va a exponer se eliminará el clasificador de estas redes y se crearán dos modelos distintos que se denominarán como:

- *modelo cam*: cuyo clasificador será una capa totalmente conectada con dos neuronas de salida. Este modelo es necesario para poder probar la técnica de **CAM**, ya que no este método no admite redes con más de una capa totalmente conectada.

- *modelo cam_pro*: cuyo clasificador constará de tres capas totalmente conectadas:

$$\#features \rightarrow \text{int} \left(\frac{0,5 + \log_2(\#features)}{2} \right) \rightarrow \text{int} \left(\frac{0,5 + \log_2(\#features)}{2} \right) \rightarrow 2$$

donde se podrán probar el resto de técnicas. El hecho de crear un clasificador con más de una capa está fundamentado en querer mostrar la potencia que aporta el método de [Grad-CAM](#), donde generaliza a su antecesor permitiendo usarse en redes con clasificadores de más de una capa totalmente conectada.

Esto dará lugar a 8 redes distintas:

* VGG-cam	* RESNET18-cam	* MOBILENET-cam	* EFFICIENT-cam
* VGG-cam_pro	* RESNET18-cam_pro	* MOBILENET-cam_pro	* EFFICIENT-cam_pro

4.3 ENTRENAMIENTO Y OPTIMIZACIÓN DE PARÁMETROS

En esta sección se comentará el proceso de entrenamiento, las técnicas y parámetros utilizados en cada una de las fases previas al testeo de los modelos. La relevancia de una buena toma de decisiones en el entrenamiento es altísima, afectando de gran manera al desempeño de los modelos en sus tareas. Por ello, esta sección es resultado de un trabajo de análisis empírico y teórico para conseguir unos modelos óptimos a partir de las ocho redes de las que se dispone.

El proceso por el que ha pasado el entrenamiento y la optimización de parámetros es el siguiente:

1. Entrenamiento (Usando el conjunto de validación como criterio de mejor modelo).
2. Establecer los parámetros de la técnica de [Smooth Grad-CAM++](#).
3. Obtener el mejor umbral para cada par (modelo, técnica) para obtener las máscaras sobre el conjunto de validación.

4.3.1 Entrenamiento

Épocas	100
Early Stopping	16
Función de Pérdida	Entropía Cruzada
Optimizador	Descenso de Gradiente Estocástico
Learning Rate	0.001
Scheduler	Factor de 0.5 cada 5 épocas

Cuadro 4: Selección de parámetros y métodos de entrenamiento

Para entrenar las ocho redes se han seguido las mismas pautas sin diferenciar entre ellos, para poder comparar bajo mismas condiciones desde el inicio. Para estudiar la localización no supervisada que brindan las técnicas de explicabilidad estudiadas, el problema se va a afrontar como un problema de clasificación, donde las etiquetas serán:

o Benigno

1 Patológico

De esta forma se podrá determinar la bondad de la localización sin la necesidad de que un patólogo tenga que determinar de manera precisa donde se encuentra el tejido patológico para etiquetar las imágenes, ahorrándole mucho tiempo. Ya que hay dos clases posibles, la función de pérdida usada para medir el error durante el entrenamiento es la [Entropía Cruzada](#). El optimizador será el [Descenso de Gradiente Estocástico](#) que tendrá asociado un scheduler del learning rate para ir reduciéndolo a medida que avanza el entrenamiento. Por lo tanto los parámetros a tunear son: el número de épocas, el early stopping⁴, el learning rate y el scheduler del learning rate.

Al utilizar un criterio de parada, el **número de épocas** se ha puesto a 100 para dar margen a los modelos y que les de tiempo a converger. Como el conjunto de datos no es grande, se espera que los modelos tarden poco en aprender de ellos. El **learning rate** se ha puesto en un inicio a 0.001, con un **scheduler** que lo reduce a la mitad cada 5 épocas: durante el desarrollo del trabajo se ha visto empíricamente como otros valores del learning rate, así como valores alejados del scheduler daban resultados mucho peores. El número de épocas sin mejorar permitidas por el **early stopping** está fijado a 16 épocas. Se puede ver el Cuadro 4 como cuadro resumen.

⁴ El early stopping es un criterio de parada en el entrenamiento que fuerza el fin del entrenamiento cuando se acarrean un número prefijado de épocas sin mejorar.

4.3.1.1 Proceso de entrenamiento

El proceso que se ha seguido en el entrenamiento es

- Por cada época:
 - **Fase de entrenamiento:** Para cada Batch de los datos de entrenamiento:
 1. Se actualizan los pesos del modelo (uso del optimizador y del scheduler del learning rate).
 2. Se recogen los valores de pérdida (uso de la métrica de pérdida) y de accuracy, para guardar un seguimiento (o track) del entrenamiento.
 - **Fase de validación:**
 1. Al final de cada paso de todos los datos de entrenamientos, se ejecutará una época de validación, comprobando la bondad del modelo sobre los datos del conjunto de validación.
 2. Se calculará igualmente la pérdida y el accuracy.
 3. Se compararán estos datos con datos previos obtenidos sobre validación.
 - Si se mejora la pérdida que la anterior mejor sobre validación, se guardarán los pesos actuales como los pesos del mejor modelo encontrado. El número de épocas sin mejorar se establecerá a cero.
 - Si no se mejora, el número de épocas sin mejorar aumenta en uno. En caso de que haya superado el early stopping se termina el modelo y cogemos el mejor modelo encontrado en validación.

4.3.1.2 Resultados del entrenamiento

Ya se han ejecutado los entrenamientos de todos los modelos. El tracking del entrenamiento se puede ver gráficamente en las Figuras 18, 19, 20, 21 y 22 (sobre el conjunto de TRAIN) y 23 (sobre el conjunto de VALIDACIÓN). Las gráficas de TRAIN se han desglosado por redes base para poder visualizarlas mejor. Con validación no se ha visto necesario puesto que lo importante era ver convergencia en los valores, y la gráfica de todos los modelos es lo suficientemente limpia para sacar conclusiones.

Las etapas de entrenamiento han variado mucho en función de la red base usada. En la Figura 22 (TRAIN) se pueden ver los tiempos de entrenamiento y los valores alcanzados. Se puede observar cómo los modelos tienen tiempos y valores muy distintos entre sí en función de la red base sobre la que se han construido. Las gráficas desglosadas en las Figuras 18, 19, 20 y 21 se ve como, independientemente del cla-

sificador, los modelos *cam* y *cam_pro*, siguen una línea similar para una misma red base.

En la [Figura 23](#) se observa cómo los modelos han conseguido converger durante su entrenamiento y que han acabado de manera propia por early stopping.

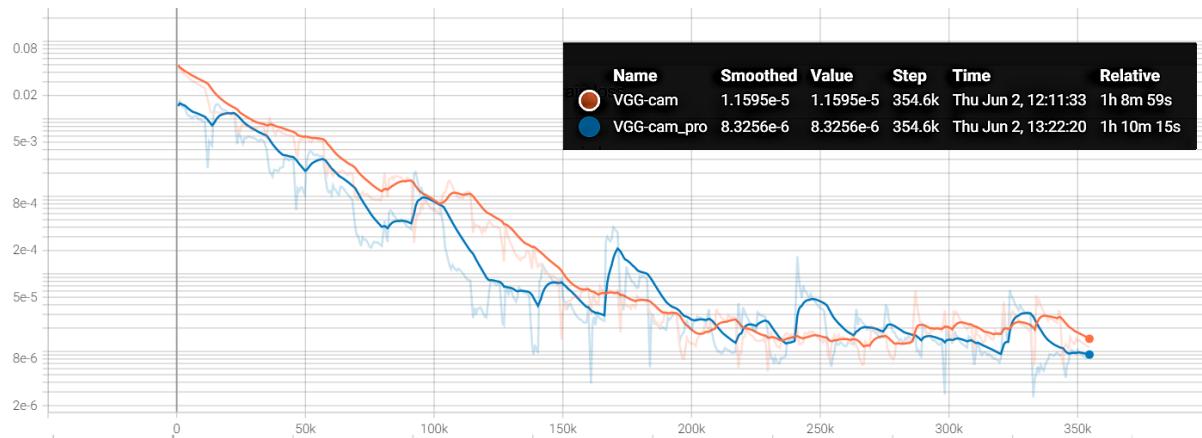


Figura 18: Track sobre los datos de train a lo largo del entrenamiento sobre los modelos de **VGG**. En el ejeY se encuentra el valor de la función de pérdida y en el ejeX los step de los batches. La gráfica se muestra con un smooth aplicado del 0.9 ofrecido por la herramienta de *tensorboard*.

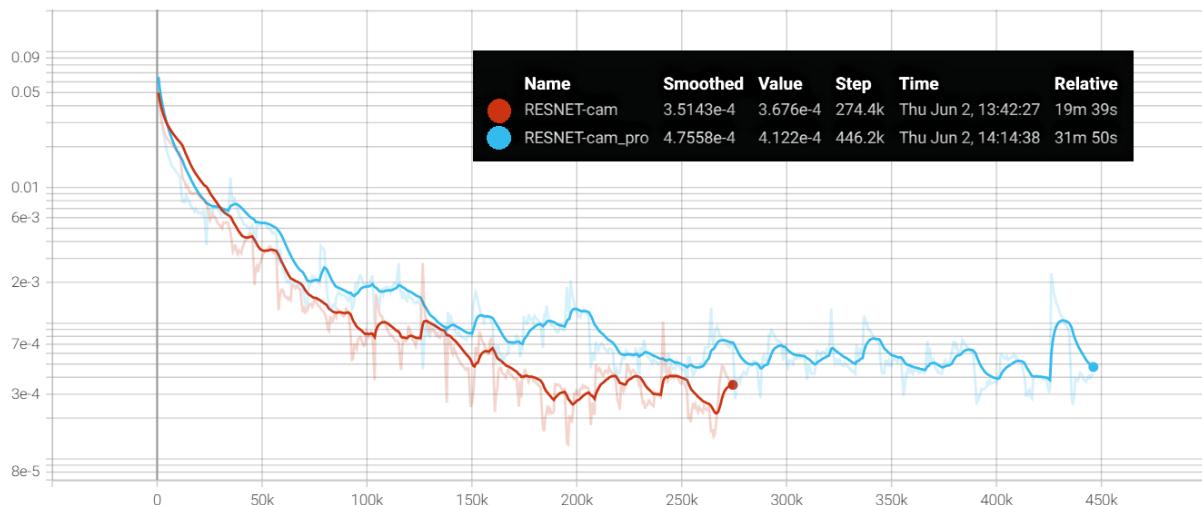


Figura 19: Track sobre los datos de train a lo largo del entrenamiento sobre los modelos de **RESNET18**. En el ejeY se encuentra el valor de la función de pérdida y en el ejeX los step de los batches. La gráfica se muestra con un smooth aplicado del 0.9 ofrecido por la herramienta de *tensorboard*.

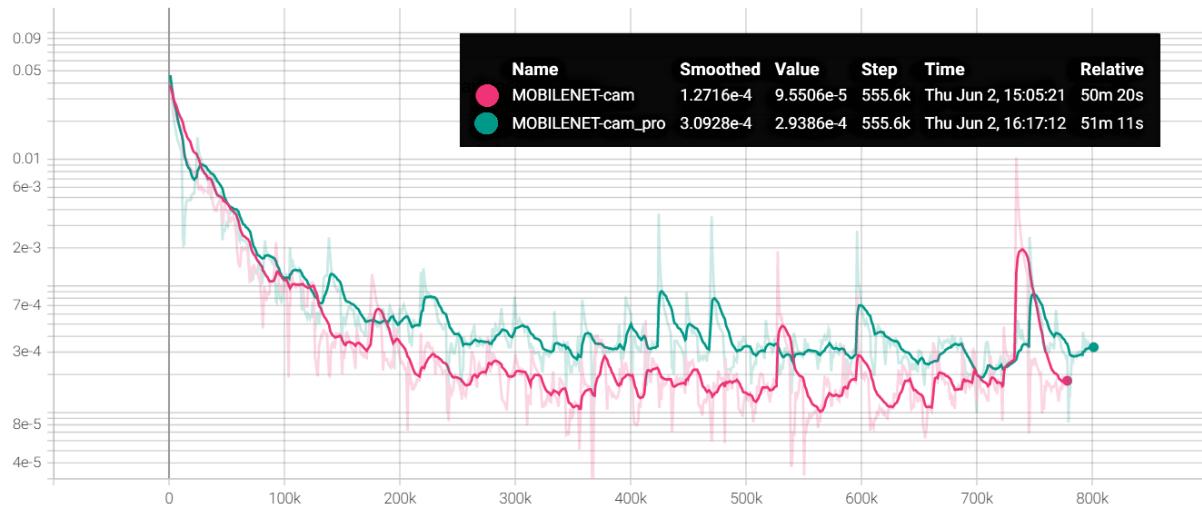


Figura 20: Track sobre los datos de train a lo largo del entrenamiento sobre los modelos de **MOBILENET**. En el ejeY se encuentra el valor de la función de pérdida y en el ejeX los step de los batches. La gráfica se muestra con un smooth aplicado del 0.9 ofrecido por la herramienta de *tensorboard*.

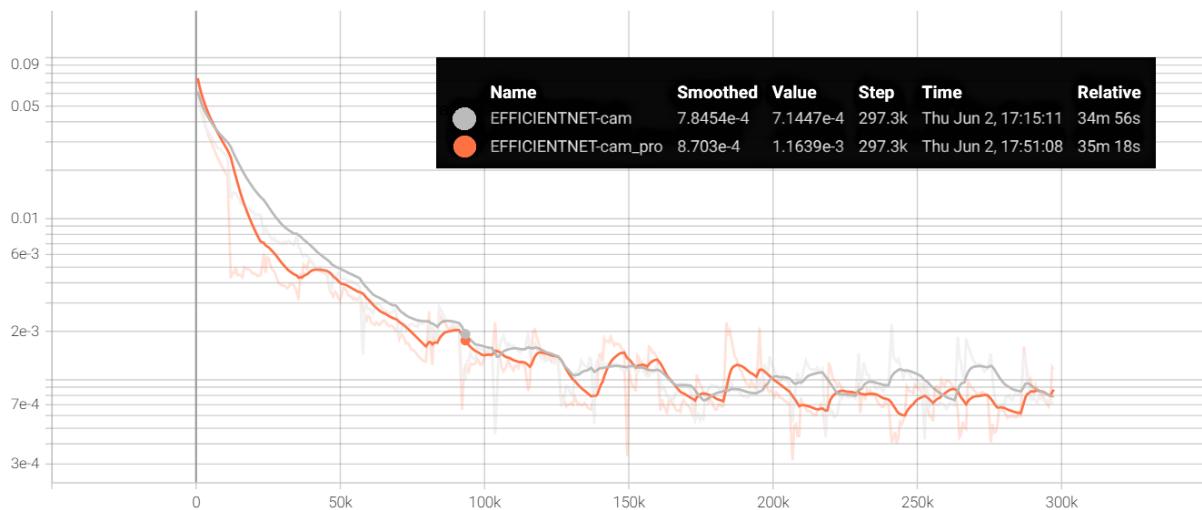


Figura 21: Track sobre los datos de train a lo largo del entrenamiento sobre los modelos de **EFFICIENTNET**. En el ejeY se encuentra el valor de la función de pérdida y en el ejeX los step de los batches. La gráfica se muestra con un smooth aplicado del 0.9 ofrecido por la herramienta de *tensorboard*.

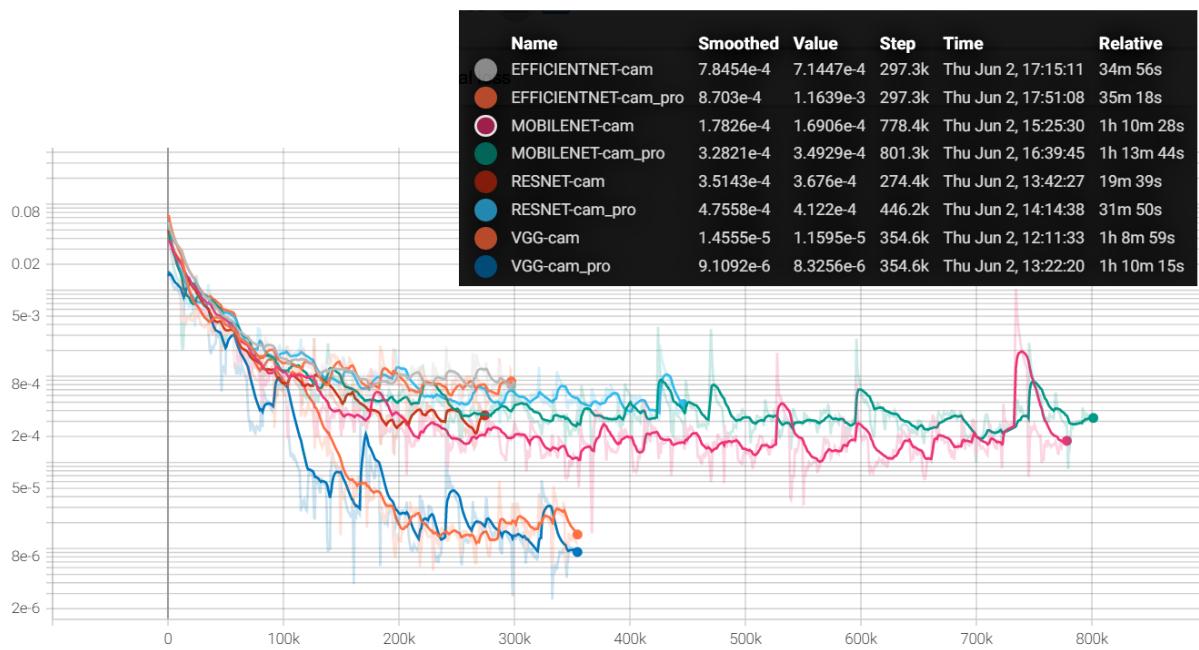


Figura 22: Track sobre los datos de train a lo largo del entrenamiento sobre **todos** los modelos. En el ejeY se encuentra el valor de la función de pérdida y en el ejeX los step de los batches. La gráfica se muestra con un smooth aplicado del 0.9 ofrecido por la herramienta de *tensorboard*.

Anticipándonos a los resultados, estas gráficas dan para imaginar qué modelo va a tener mejor desempeño que el resto y qué modelo va a ser el peor. Como peor modelo se puede pensar en los modelos de *EFFICIENTNET*: han sido los modelos que menos épocas han entrenado (junto con *RESNET18-cam*), los primeros en encontrar un entorno muy cercano a un mínimo local y esto probablemente haya impedido hacer una exploración para encontrar mínimos locales más pequeños, ya que los valores de pérdidas de los modelos de *EFFICIENTNET* son los más altos de entre todos los modelos.

No pasa así con *RESNET18-cam*, que a pesar de tardar poco en su entrenamiento, consigue valores de pérdida menores, acorde a su modelo hermano *RESNET18-cam_pro* que si ha necesitado bastante más épocas en terminar el entrenamiento.

MOBILENET si ha conseguido desempeños similares con épocas similares entre sus modelos, siendo los dos modelos que más épocas, con diferencia, han entrenado. En su gráfica de validación (Figura 23), se ve una convergencia muy estable desde una época mucho más temprana a su época final. Cabe pensar que el modelo ya había conseguido valores lo suficientemente buenos mucho antes de terminar el proceso de entrenamiento completo.

Por último, *VGG* consigue las mejores pérdidas en el conjunto de entrenamiento, pero las peores en validación (Figura 23) aunque siguen siendo valores realmente buenos.

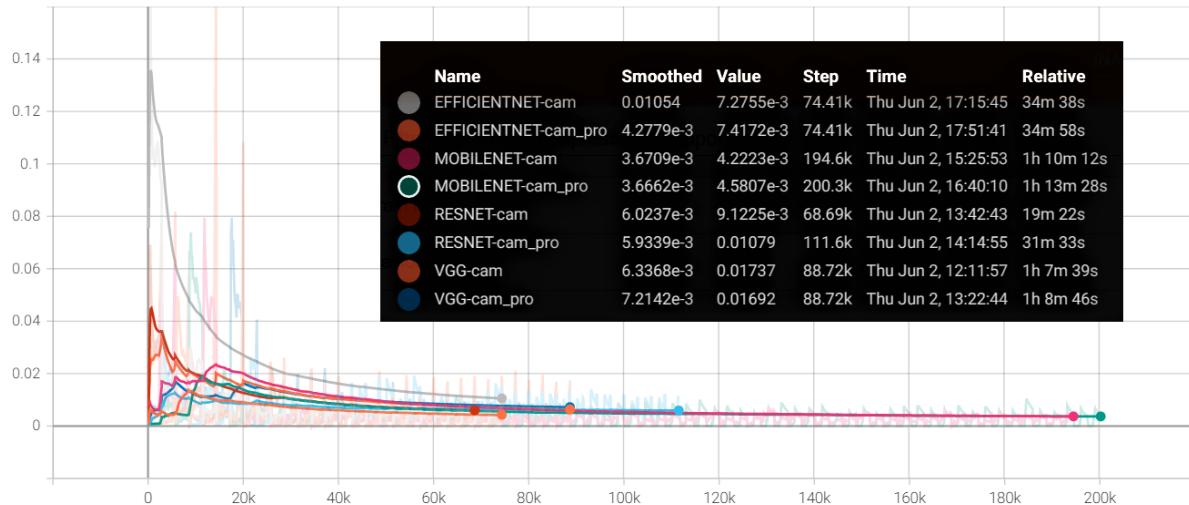


Figura 23: Track sobre los datos de validación a lo largo del entrenamiento sobre **todos** los modelos. En el ejeY se encuentra el valor de la función de pérdida y en el ejeX los step de los batches. La gráfica se muestra con un smooth aplicado del 0.999 ofrecido por la herramienta de *tensorboard*.

VGG		RESNET18		MOBILENET		EFFICIENTNET	
cam	cam_pro	cam	cam_pro	cam	cam_pro	cam	cam_pro
1h 7m 39s	1h 8m 46s	19m 22s	31m 33s	1h 10m 12s	1h 13m 28s	34m 38s	34m 58s

Cuadro 5: Tiempos de entrenamiento. Datos del ordenador sobre el que se ha ejecutado: RAM 16GB; Gráfica NVIDIA GeForce RTX3070 Laptop GPU; Procesador AMD Ryzen 9 5900HX with Radeon Graphics, 3301 Mhz, 8 procesadores principales, 16 procesadores lógicos.

En cuanto a los tiempos⁵ de entrenamiento se tiene una comparación en formato tabla en el [Cuadro 24](#) y en formato gráfica en la [Figura 24](#). Aquí solamente comentar la influencia del número de parámetros y de la profundidad que posee cada red. Si se recuerda los datos del [Cuadro 3](#), se ve como el modelo con más cantidad de parámetros, VGG, es el que tiene una relación tiempo/época más alta. MOBILENET refleja tiempos altos porque ha durado muchas épocas. RESNET18 y EFFICIENTNET mantienen tiempos similares, ya que, a pesar de que EFFICIENTNET tiene menos parámetros, tiene una profundidad mucho mayor, lo que requiere tiempo de cómputo de los gradientes.

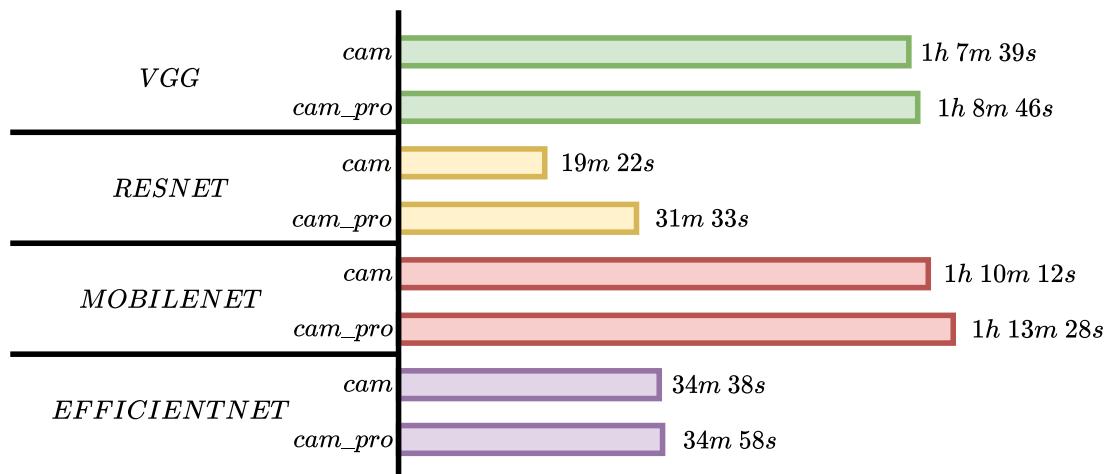


Figura 24: Tiempos entrenamiento. Datos del ordenador sobre el que se ha ejecutado: RAM 16GB; Gráfica NVIDIA GeForce RTX3070 Laptop GPU; Procesador AMD Ryzen 9 5900HX with Radeon Graphics, 3301 Mhz, 8 procesadores principales, 16 procesadores lógicos.

4.3.2 Optimización de parámetros de Smooth Grad-CAM++

Los parámetros que se han usado para la técnica de [Smooth Grad-CAM++](#) se han cogido en base a las recomendaciones de Omeiza et al. [5]: se generarán 10 imágenes con un ruido generado por una desviación típica de 0.25. Los autores recomendaban que las imágenes con ruido generadas fueran imperceptibles para el ojo humano. Con este consejo unido a pruebas empíricas con valores entre 0.1 y 0.5, se encontró que, entre los valores estudiados, el que mejor respondía era un valor de 0.25 para la desviación típica. El número de imágenes es 10 para tener una muestra considerable para ver cambios y no tan grande como para alargar mucho las pruebas. El tiempo de ejecución de la técnica de [Smooth Grad-CAM++](#) con respecto a las otras es $\mathcal{O}(N)$,

⁵ NOTA: Los datos de tiempo vienen determinados por el tipo de ordenador usado. Dentro de cada gráfico o tabla donde se muestren estadísticas de los tiempos se incorpora un resumen de los componentes del ordenador que lo ha ejecutado.

donde N es el número de imágenes a procesar. Es decir, con un $N = 10$, el tiempo que tarda en ejecutarse una imagen con **Smooth Grad-CAM++**, se podrían ejecutar 10 imágenes con **Grad-CAM++**

N	10	σ	0.25
-----	----	----------	------

Cuadro 6: Parámetros de **Smooth Grad-CAM++** usados en el trabajo.

4.3.3 Optimización de umbral de máscara

Este paso es únicamente útil para ocasiones en las que se dispongan de datos etiquetados con localización. El proceso de creación de una máscara a partir de un mapa de calor consiste en discretizar el mapa de calor con un umbral. Una vez que se tiene la máscara generada se podrá comparar con la máscara original con el cálculo del **IoU**, métrica utilizada para medir la bondad de la localización.

Para obtener el mejor umbral se ha desarrollado un proceso donde primero se normaliza el mapa de calor restándole su mínimo y dividiendo por su valor máximo (esto nos lleva los valores dentro de $[0, 1]$ ²²⁴). Se hace una búsqueda del mejor umbral sobre el intervalo $[0, 1]$. Concretamente se ha buscado en los valores:

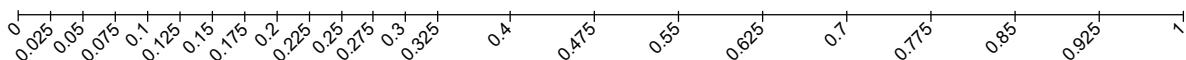


Figura 25: Valores del umbral de máscara investigados.

En la [Figura 26](#) se muestran las gráficas para cada modelo, para cada técnica.

	CAM	Grad-CAM	Grad-CAM++	Smooth Grad-CAM++
VGG	0.075	0.05	0.05	0.05
RESNET18	0.075	0.05	0.05	0.075
MOBILENET	0.05	0.05	0.05	0.05
EFFICIENTNET	0.000	0.025	0.025	0.05

Cuadro 7: Mejores umbrales encontrados para cada técnica en cada modelo.

El *mejor valor IoU* obtenido en cada par (modelo,técnica) habla de cómo de bien podría localizar el modelo con un umbral óptimo. Esta métrica permite hacer comparaciones entre cualquier modelo y técnica.

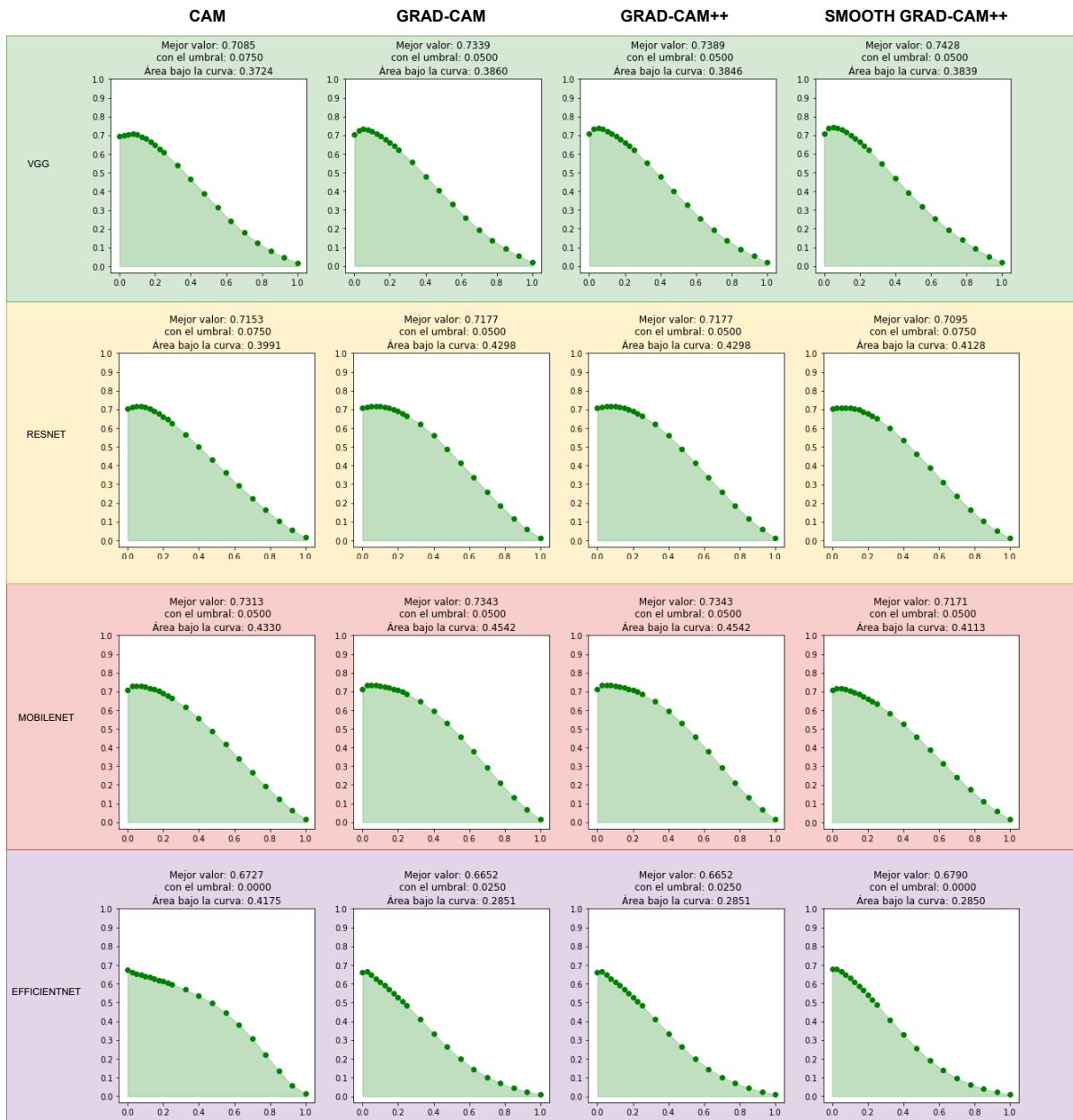


Figura 26: IoU (eje Y) estudiado sobre las máscaras de las imágenes del conjunto de **validación**, con los umbrales (eje X) de la Figura 25 para obtener el mejor umbral. Notar que en la técnica de **CAM** se ha usado el modelo *red_cam*, mientras que en el resto de técnicas se ha usado el modelo *red_cam_pro* para cada red: VGG, RESNET18, MOBILENET o EFFICIENTNET.

4.4 ANÁLISIS DE MÉTRICAS SOBRE TEST

Con todo el proceso de entrenamiento realizado se llega a la etapa de **test**, donde se pondrán a prueba todos los modelos con las técnicas de explicabilidad frente al conjunto de datos de test, que no ha sido visto con anterioridad.

En esta sección se medirá la bondad de la clasificación (para lo que han sido entrenados los modelos) primero, ayudando al análisis posterior de localización. Al ser modelos que han sido entrenados de manera débilmente supervisada en localización, el estudio de sus métricas de clasificación pueden dar pistas de la calidad de la localización obtenida.

4.4.1 Métricas de clasificación

Para medir la bondad de la localización se hará uso de las métricas de:

- *Accuracy*: dará información sobre el porcentaje de casos que el modelo ha acertado.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

- *Recall*: es la proporción de casos positivos que el modelo ha sido capaz de identificar.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- *Precision*: permite medir cuántos casos positivos dados por el modelo, son realmente casos positivos.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- *F1-Score*: métrica que combina las métricas de precision y recall.

$$F1 - Score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Donde la clase positiva será la patológica. Todas estas métricas cogen valores en el intervalo $[0,1]$. Cuanto más cercana del 1 esté el valor de la métrica, mejor se comportará el modelo según ella. En tareas como la detección de cáncer, es importante que el modelo tenga un *recall* alto, ya que un positivo dará lugar a seguir al paciente y hacer más pruebas, ayudando a confirmar ese resultado positivo.

	VGG		RESNET18		MOBILENET		EFFICIENTNET	
	cam	cam_pro	cam	cam_pro	cam	cam_pro	cam	cam_pro
Accuracy	0.9862	0.9912	0.9791	0.9826	0.9813	0.9820	0.9625	0.9621
Recall	0.9792	0.9867	0.9633	0.966	0.9774	0.9714	0.9503	0.9518
Precision	0.9903	0.9939	0.9904	0.9953	0.9813	0.9887	0.9663	0.964
F1-Score	0.98471	0.9903	0.9767	0.9804	0.9793	0.98	0.9582	0.9579

Cuadro 8: Métricas de clasificación de cada modelo. Las celdas destacadas en verde resaltan la mejor marca de los modelos en cada métrica.

El modelo que mejor clasifica es *VGG cam_pro* (Cuadro 8). Posee las métricas más altas exceptuando la *precisión* (donde el mejor es *RESNET18 cam_pro*). Como se ha comentado, para diagnóstico de enfermedades la métrica de *recall* tiene gran importancia y aquí, *VGG cam_pro* es mejor. Además, el *F1-Score* dice que en la compensación *recall-precision* gana *VGG cam_pro*, dejándolo como claro vencedor.

A pesar de ello, todos los modelos clasifican bastante bien, exceptuando *EFFICIENTNET*, el cual tiene unas métricas bastante más bajas que el resto.

Estos resultados ya se podían intuir tras el entrenamiento, donde los modelos de *EFFICIENTNET* eran los que mayor pérdida tenían (quizás consecuencia de ser los modelos que menos épocas lograron entrenar).

VGG por su parte, conseguía las mejores pérdidas en los datos de entrenamiento, pero de las peores en validación (aunque seguían siendo bajas). Viendo ahora como uno de sus modelos es el vencedor, se puede decir que el modelo se ajustó a los datos de entrenamiento consiguiendo generalizar muy bien. Esto quiere decir que supo encontrar los patrones asociados a imágenes patológicas (positivas).

Por último, los modelos de *RESNET18* y *MOBILENET* tienen bastante buenos resultados de clasificación en todas sus métricas, bastante más cerca del mejor modelo que del peor.

4.4.2 Métricas de localización

En esta sección se presentará el estudio de las métricas de localización. El estudio se realizará sobre imágenes patológicas únicamente, ya que solo hay interés de localización cuando la imagen se ha clasificado como tejido patológico.

Para llevar el estudio a cabo se han usado métricas parecidas a las de clasificación pero aplicadas a los píxeles. Dada una máscara, esta discriminará a los píxeles en dos clases: positiva (hay tejido patológico) o negativa (no hay tejido patológico). De esta manera para cada imagen se podrá generar una matriz de confusión (TP, FP, TN, FN) permitiendo el estudio de métricas como:

- *IoU*: El *intersection over union* se puede ver en términos de matriz de confusión.

$$IoU = \frac{TP}{TP + FP + FN}$$

- *Recall*: en localización se puede ver visualmente como la proporción de área que hay de la intersección de las máscaras dentro de la máscara **original**.

$$Recall = \frac{TP}{TP + FN}$$

- *Precision*: en localización se puede ver visualmente como la proporción de área que hay de la intersección de las máscaras dentro de la máscara **predicha**.

$$Precision = \frac{TP}{TP + FP}$$

- *F1-Score*. En términos de localización es una medida muy similar al *intersection over union*:

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

El interés del *F1-Score* está en medir el equilibrio entre el *recall* y la *precision*. Así, aunque sea muy similar al *IoU*, se va a estudiar su valor.

Estas métricas se calculan por imagen, por lo que hará falta hacer una media del conjunto de test para poder obtener resultados globales.

4.4.2.1 Estudio de las máscaras de localización

Para poder estudiar las métricas anteriormente nombradas es necesario tener una máscara asociada al mapa de calor del modelo. Para ello se va a umbralizar de la misma manera que se hizo en la etapa de entrenamiento para buscar el mejor umbral: se crearán máscaras para cada valor del umbral de la [Figura 25](#). Se seleccionará el mejor umbral y en base a la máscara generada por ese umbral se calcularán las métricas.

4.4.2.2 Selección de datos a estudiar

Como se puede observar en la gráfica de la [Figura 27](#) hay una gran proporción de imágenes cuyo tejido cancerígeno es el total de la imagen o gran parte de la imagen. Gracias a esto, una máscara trivial (asume que el tejido cancerígeno es el 100 % de la imagen) provocará en el *IoU* lo siguiente:

$$IoU_{Mask_trivial} = \frac{Mask_Original \cap Mask_Predicha}{Mask_Original \cup Mask_Predicha} = \frac{Mask_Original}{Mask_Predicha} = Mask_Original$$

Donde se ha usado en la primera igualdad que la máscara original está incluída dentro de la predicha y en la última igualdad que el tejido cancerígeno de la máscara predicha ocupa el 100 % (es decir un 1.). De esta manera el IoU será la media del porcentaje tejido cancerígeno que hay en las imágenes de test. La media del porcentaje de tejido cancerígeno que hay en las imágenes de test es 0.66, por lo tanto:

$$IoU_{Mask_trivial} = 0,66$$

siendo un valor de IoU bastante alto a priori, pero que realmente no da significado, puesto que es una máscara trivial.

Este buenhacer de la máscara trivial provoca que los modelos encuentren su mejor umbral en un nivel muy bajo. Además, estos umbrales bajos pueden llegar a funcionar bien sobre imágenes que no contengan tanto tejido cancerígeno ya que su *Recall* hará que en las métricas de IoU y *F1-Score* consigan valores aceptables compensando el valor bajo de *Precision*. Por ello se ha decidido coger solo aquellas imágenes que posean menos de un 50 % de tejido cancerígeno del total de su imagen. El número de imágenes que entran dentro de este subconjunto de test es 1084, suficiente para poder sacar conclusiones.

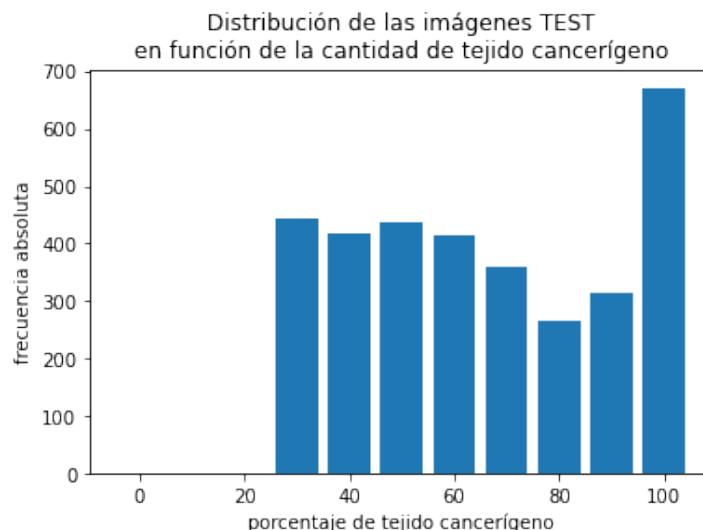


Figura 27: Distribución de las imágenes de test patológicas en función del porcentaje de tejido cancerígeno que poseen.

El proceso de búsqueda de mejor umbral junto con el cálculo del resto de métricas se ve reflejado en las Figuras 29, 30, 31 y 32.

4.4.3 Comparación de modelos y resultados

Definidas las métricas de comparación y análisis, en esta parte del trabajo se van a comparar todos los modelos y todas las técnicas entre sí, se sacará el mejor par (modelo, técnica) y se verá su bondad sobre el problema.

La calidad de la solución quedará marcada bajo el subconjunto de test donde el tejido cancerígeno sea como mucho el 50 % de la propia imagen.

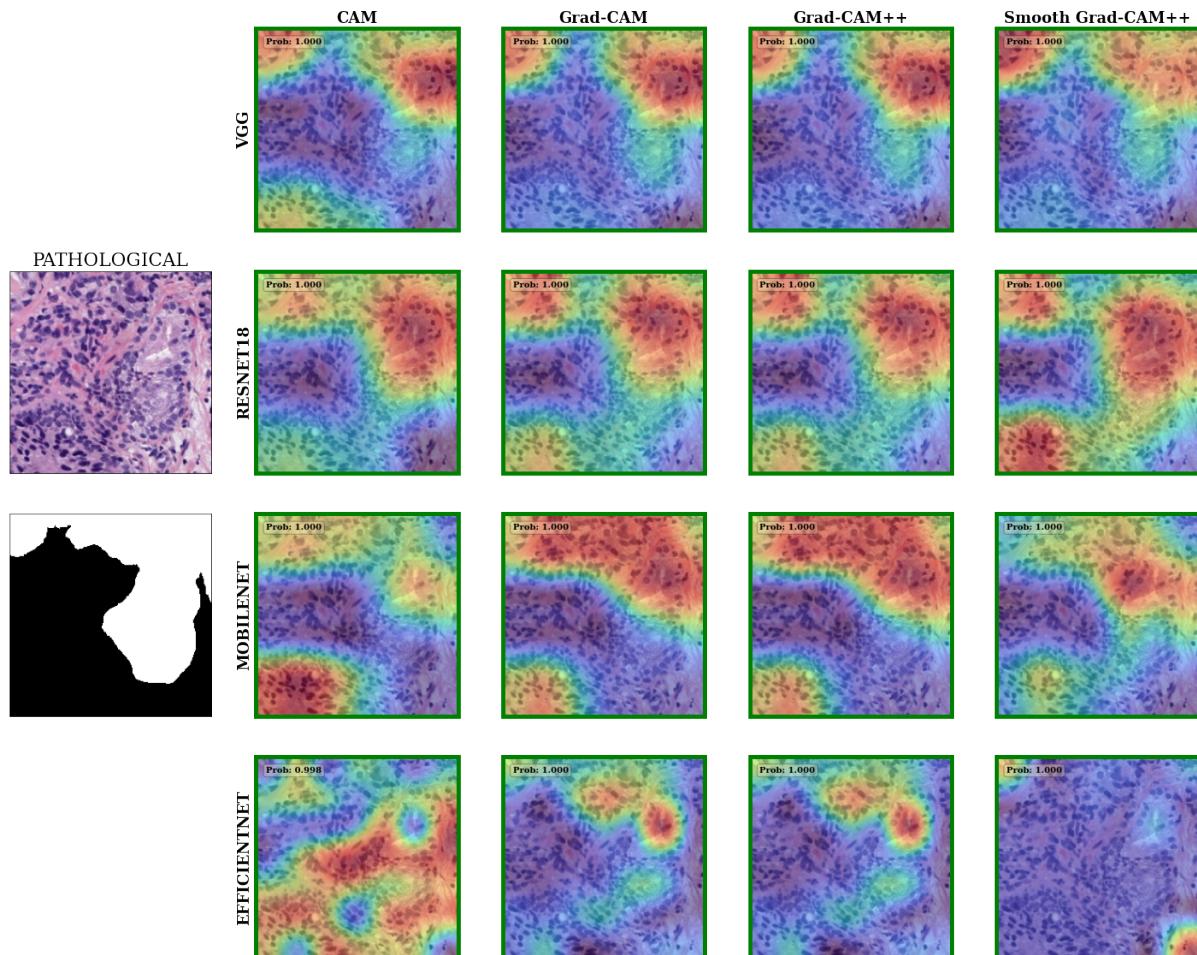


Figura 28: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. En este ejemplo los modelos de *VGG*, *RESNET*, *MOBILENET* captan muy bien la parte superior de la imagen donde hay tejido cancerígeno. Sin embargo, añaden en la esquina inferior izquierda una región donde dicen que hay tejido cancerígeno(en mayor o menor medida, dependiendo del modelo y de la técnica). *EFFICIENTNET* por su parte, localiza muy mal con *CAM* y *smooth Grad-CAM++*, mejorando con *Grad-CAM* y *Grad-CAM++*, seleccionando bien la parte superior. En el [Apéndice A](#) se pueden ver más ejemplos.

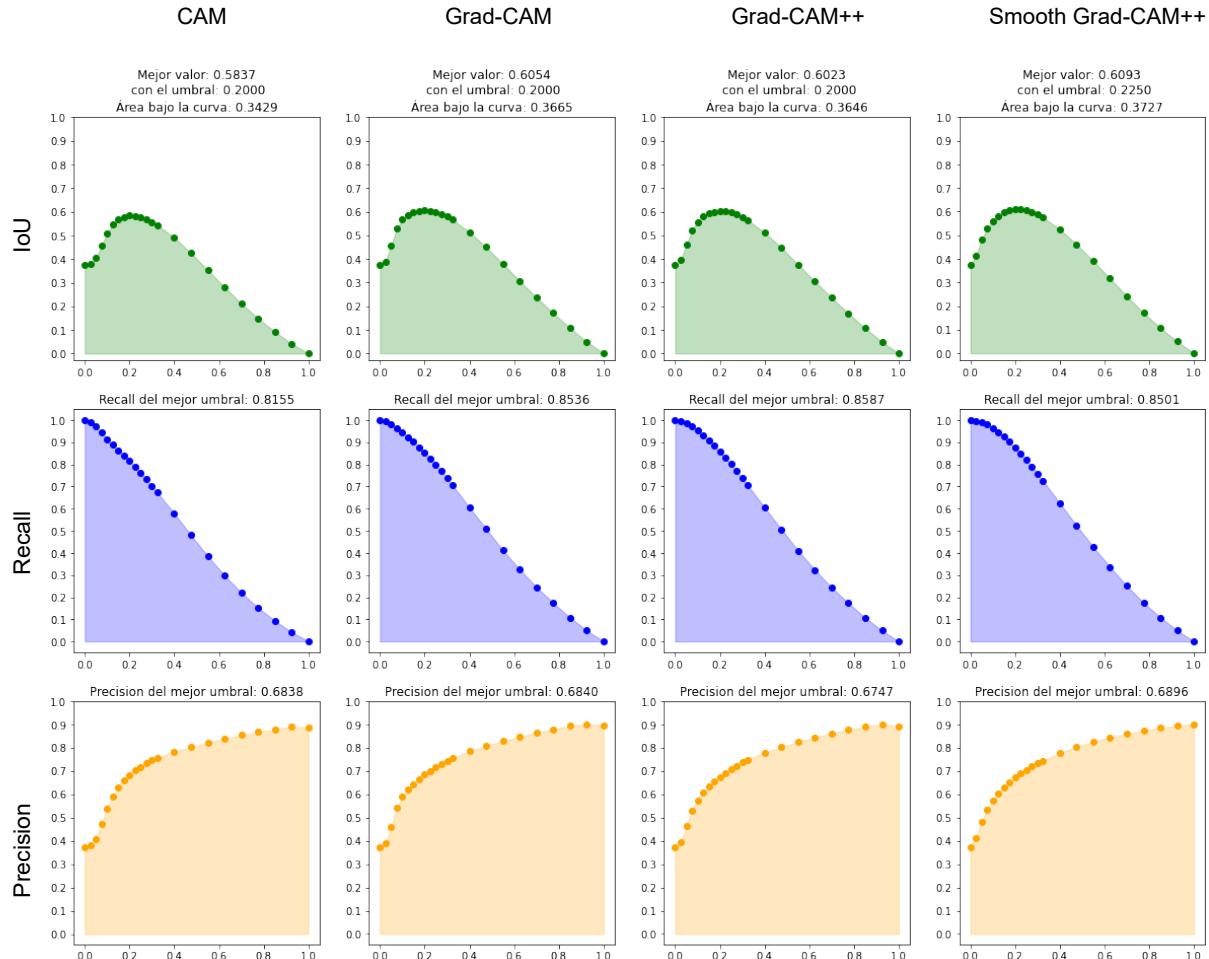


Figura 29: Modelo VGG: Métricas de localización en test estudiadas sobre las máscaras con los umbrales (eje X) de la [Figura 25](#). En verde se encuentra el estudio del Intersection over Union, en azul el Recall y en naranja la Precisión.

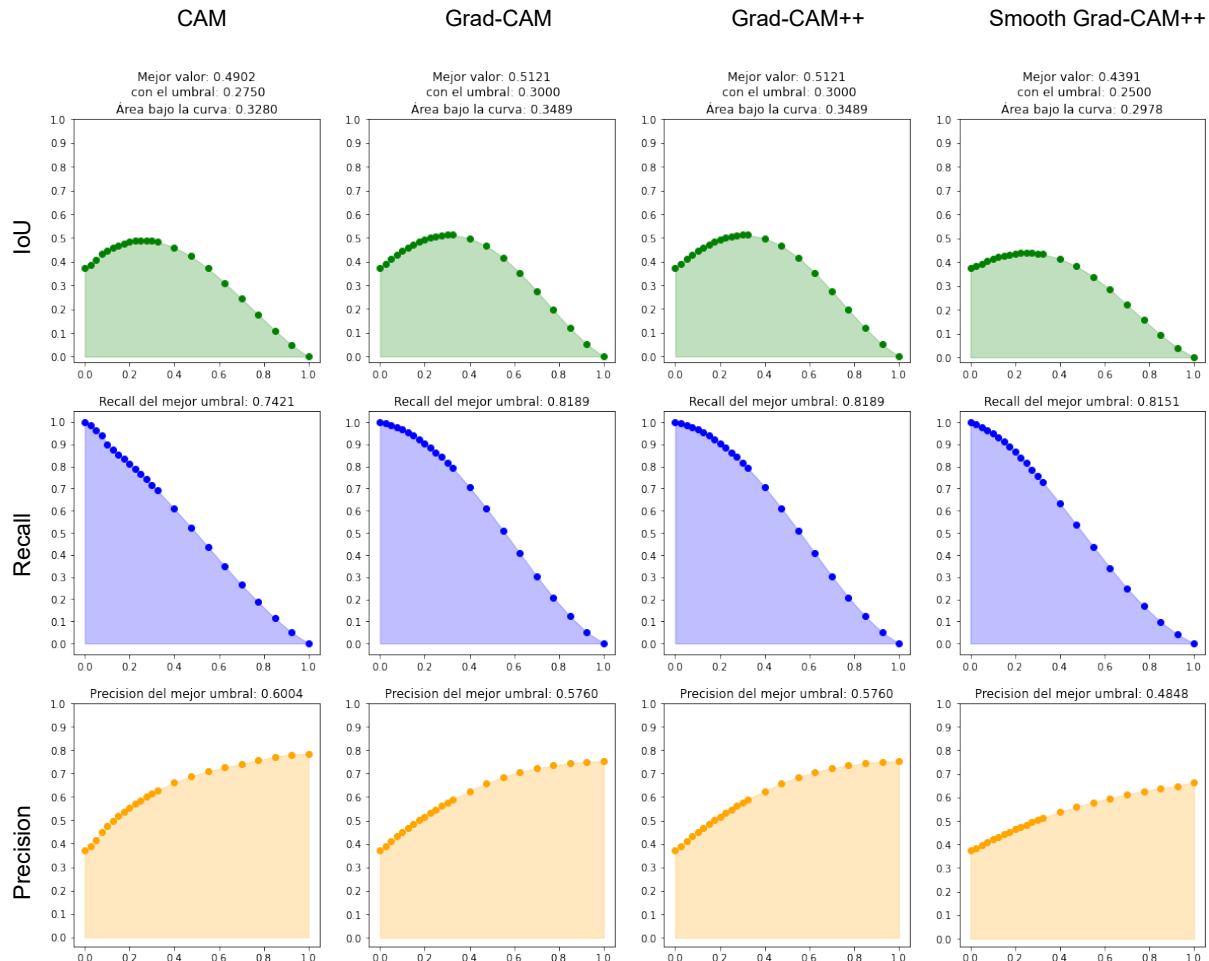


Figura 30: Modelo RESNET18: Métricas de localización en test estudiadas sobre las máscaras con los umbrales (eje X) de la [Figura 25](#). En verde se encuentra el estudio del Intersection over Union, en azul el Recall y en naranja la Precisión.

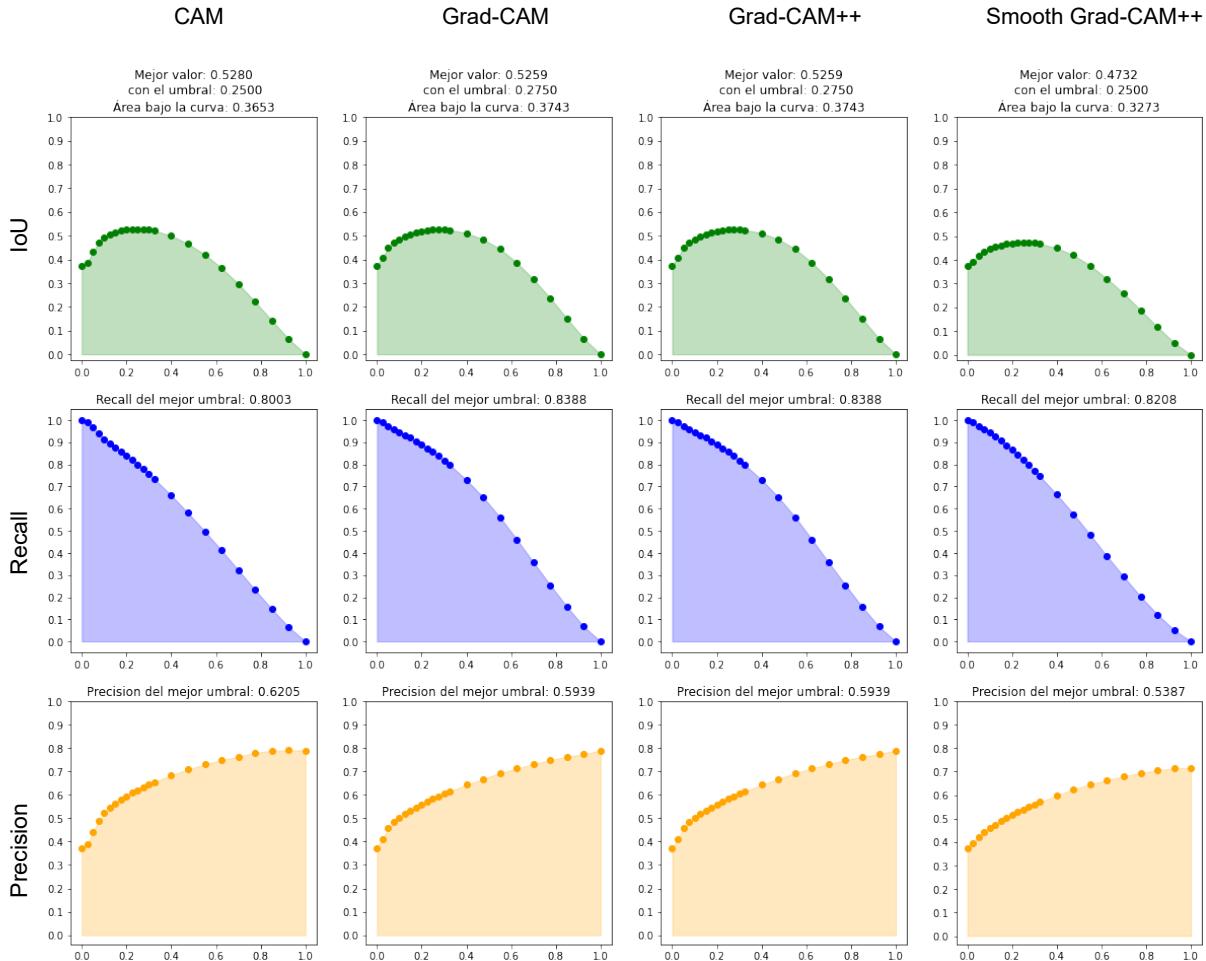


Figura 31: Modelo MOBILENET: Métricas de localización en test estudiadas sobre las máscaras con los umbrales (eje X) de la [Figura 25](#). En verde se encuentra el estudio del Intersection over Union, en azul el Recall y en naranja la Precisión.

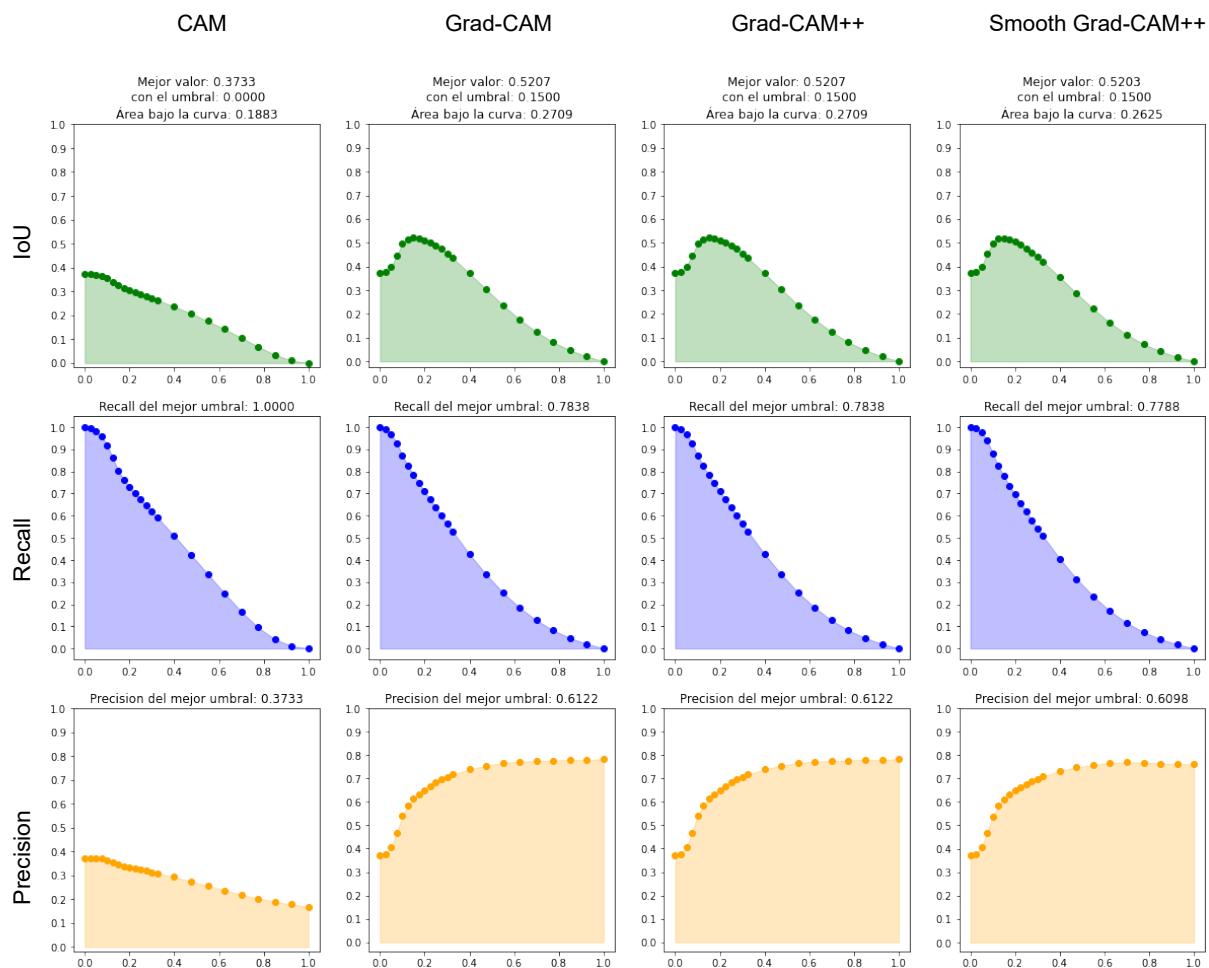


Figura 32: Modelo EFFICIENTNET: Métricas de localización en test estudiadas sobre las máscaras con los umbrales (eje X) de la Figura 25. En verde se encuentra el estudio del Intersection over Union, en azul el Recall y en naranja la Precisión.

Para interpretar las gráficas de las Figuras 29, 30, 31 y 32 se van a dar algunas directrices. El primero es acerca del umbral, conforme el umbral crece, la máscara generada se va restringiendo más y más, pasando de la máscara completa a la máscara vacía.

Las gráficas del *IoU* (color verde) van a dar una idea de cómo el modelo discrimina la localización entre clases en función de la forma de la curva:

C1 Cuanto más plana sea la cima de la curva mejor separa el modelo lo que él cree que es clase positiva y lo que no.

C2 Cuanto mayor sea la altura de la cima mejor sabe el modelo localizar la clase positiva.

Por otro lado las gráficas de *Recall* (color azul) van a dar información sobre cómo de bien contiene el mapa de calor generado a la máscara original.

C3 Es una curva decreciente.

C4 Esta curva siempre empieza en 1 (*umbral* = 0) y acaba en 0 (*umbral* = 1).

C5 La curva sería mejor cuanto más umbrales se mantuviesen cerca del 1.

Por último, las gráficas de *Precision* (color naranja) dan información sobre cómo de bien la máscara original contiene al mapa de calor.

C6 No tiene porqué ser monótona.

C7 Una curva que refleja un buen comportamiento sería creciente: esto significaría que el mapa de calor se ha ido centrando cada vez en mayor proporción en zonas donde realmente hay tejido cancerígeno. Si en su último umbral es un 1 significa que el mapa de calor ha destacado como zona más importante una zona que pertenece a la máscara original.

C8 Una curva decreciente refleja que el modelo no ha sabido localizar en ningún momento la clase positiva: se ha centrado en zonas que no pertenecen a la clase positiva.

C9 Su valor inicial va a ser la media de los porcentajes de tejido cancerígeno que tienen las imágenes sobre las que estamos trabajando.

	VGG				RESNET18				MOBILENET				EFFICIENTNET			
	cam		cam_pro		cam		cam_pro		cam		cam_pro		cam		cam_pro	
	C	GC	GC++	SGC++	C	GC	GC++	SGC++	C	GC	GC++	SGC++	C	GC	GC++	SGC++
Umbral	0.2	0.2	0.2	0.225	0.275	0.3	0.3	0.25	0.25	0.275	0.275	0.275	0.	0.15	0.15	0.15
IoU	0.5837	0.6054	0.6023	0.6093	0.4902	0.5121	0.5121	0.4391	0.528	0.5259	0.5259	0.4732	0.3733	0.5207	0.5207	0.5203
Recall	0.8155	0.8536	0.8587	0.8501	0.7421	0.8189	0.8189	0.8151	0.8003	0.8388	0.8388	0.8208	1.	0.7838	0.7838	0.7788
Precision	0.6838	0.684	0.6747	0.6896	0.6004	0.5760	0.5760	0.4848	0.6205	0.5939	0.5939	0.5387	0.3733	0.6122	0.6122	0.6098
F1-Score	0.7439	0.7594	0.7557	0.7615	0.6638	0.6763	0.6763	0.6175	0.699	0.694	0.694	0.6487	0.5437	0.6875	0.6875	0.684

Cuadro 9: Métricas de localización de cada modelo para cada técnica (**C**: CAM, **GC**: Grad-CAM, **GC++**: Grad-CAM++, **SGC++**: Smooth Grad-CAM++) con el subconjunto de datos test que poseen un 50 % o menos de tejido cancerígeno. La celda destacada en verde resalta la mejor marca de entre todos los modelos y técnicas. Correspondiente a las Figuras 29, 30, 31 y 32.

4.4.3.1 Cota superior de IoU de localización

No hay que olvidar que los métodos de explicabilidad que se están usando, redimensionan la salida del extractor de características al tamaño original de la imagen. Esta redimensión provoca que nunca se pueda llegar a un $IoU=1$. Para establecer un

valor que se aproxime a lo que sería una cota superior se han redimensionado las máscaras originales a tamaño 7×7 , las dimensiones en las que se obtienen los mapas de calor. Esta redimensión se hace a través de una interpolación bicúbica a través de la funcionalidad ofrecida por *OpenCV*: `cv2.INTER_CUBIC`. Posteriormente se ha vuelto a redimensionar a su tamaño original, y hecho esto se han empezado a estudiar los umbrales, como se harían con unas máscaras que se han sacado de un modelo. El resultado se puede ver en la [Figura 33](#).

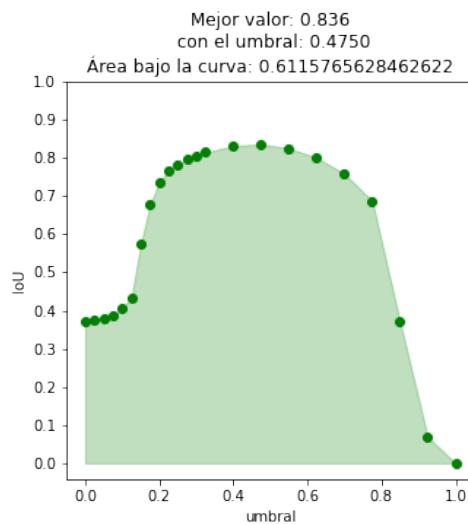


Figura 33: Estudio de umbralización con las máscaras originales, redimensionándolas primero a tamaño 7×7 para volver a redimensionarlas a tamaño original.

Así, se puede decir que una buena aproximación a lo que sería una cota superior sería:

$$IoU_{sup} = 0,836 \quad (31)$$

4.4.3.2 Comparación de modelos

Poniendo ahora la vista en el [Cuadro 9](#), se observa como los umbrales tienen sentido, exceptuando el modelo *EFFICIENTNET cam*, con umbral cero (muy mal indicador).

Estudiando el valor *IoU* se va a establecer un ranking con el subconjunto actual de test que estamos trabajando. El ranking sería el siguiente:

1. **VGG cam_pro** (*mejor marca*: 0.6093)
2. **VGG cam** (*mejor marca*: 0.5837)
3. **MOBILENET cam** (*mejor marca*: 0.528)

4. **MOBILENET cam_pro** (*mejor marca: 0.5259*)
5. **EFFICIENTNET cam_pro** (*mejor marca: 0.5207*)
6. **RESNET18 cam_pro** (*mejor marca: 0.5121*)
7. **RESNET18 cam** (*mejor marca: 0.4902*)
8. **EFFICIENTNET cam** (*mejor marca: 0.3733*)

además, si se hace el estudio de los modelos *cam_pro* fijando cada una de las tres técnicas se obtiene:

- para las técnicas *Grad-CAM* y *Grad-CAM++* el ranking anterior se mantiene.
- para *Smooth Grad-CAM++* el modelo *EFFICIENTNET cam_pro* asciende a segunda posición, quedando el resto igual.

Así, en términos de *IoU* se puede concluir que el mejor modelo es **VGG** independientemente de la técnica.

Análisis de localización basado en métricas de clasificación

Para intentar dar una explicación al ranking hay que recordar que esta localización se ha aprendido de manera débilmente supervisada, enseñando a los modelos únicamente a clasificar. Por ello, su calidad de clasificación va a influir notablemente en su calidad de localización. La métrica de clasificación que más relevante va a ser es el *Recall* ya que da información sobre la eficacia en decir que una imagen es patológica cuando realmente es patológica (y se está estudiando la localización únicamente sobre imágenes patológicas). Yendo al [Cuadro 8](#) se puede observar en la fila de *Recall* que si se ordenan por puntuación los modelos, el ranking que se obtendría sería muy similar al ranking que ha salido ahora, con la excepción de *EFFICIENTNET cam_pro*, que clasifica mucho peor de lo que localiza (relativamente).

Análisis de localización basado en métricas de localización

Por otra parte, se tienen que analizar las gráficas de las Figuras [29](#), [30](#), [31](#) y [32](#) teniendo en cuenta los criterios de la [Sección 4.4.3](#) que se han comentado. Comentando de manera general las gráficas entre modelos se puede decir que, usando el criterio **C1**, los modelos que mejor discriminan lo que ellos piensan que es tejido cancerígeno son *MOBILENET* y *RESNET18*; siendo *EFFICIENTNET* el peor.

En cuanto a la gráfica de *Recall* las gráficas de *EFFICIENTNET* tienen una primera bajada más fuerte, lo que hace que empeore mucho el modelo, esto significa que los

mapas de calor rápidamente dejan de cubrir parte del tejido cancerígeno real. En el resto, los modelos *cam_pro* suelen tener algo de mejoría con respecto a los *cam* de su misma red base. Esto puede ser la razón de por qué los modelos *cam_pro* se comportan mejor que los demás.

La gráfica de *Recall* siempre hay que acompañarla de la de *Precision*. Como se ha comentado, los modelos de *EFFICIENTNET* tenían las peores gráficas de *Recall*. A pesar de ello, *EFFICIENTNET cam_pro* es mejor que los modelos de *RESNET18*, pues esto es gracias a su *Precision*. En las tres técnicas del modelo se puede ver como hay una subida muy pronunciada al inicio (criterio **C7**), provocando esta compensación. En la otra cara de la moneda está *EFFICIENTNET cam* que tiene una gráfica de *Precision* descendente, algo que es muy mala señal y que quiere decir que no ha sabido localizar la clase positiva (criterio **C8**). En esta gráfica de *Precision* en *VGG* y *MOBILENET* también tienen subidas pronunciadas lo que implica un buen comportamiento (criterio **C7**). Además en *VGG* se alcanza en el último umbral el valor más alto de todos los modelos llegando a valores cercanos al 0.9, esto implica que las zonas más relevantes del modelo son zonas con tejido cancerígeno con una probabilidad de alrededor del 0.9, (criterio **C7**).

De hecho, el ranking dado por las mejores marcas de *intersection over union* dejan un orden que podría sacarse con los criterios de la [Sección 4.4.3](#) fijándose únicamente en las gráficas de *Recall* y *Precision*:

1. **VGG cam_pro:** el hecho de que las gráficas de *Recall* son muy parecidas en forma a las de los modelos de *RESNET18* y *MOBILENET* y con una mejor forma que los modelos de *EFFICIENTNET*; unido al hecho de tener una pendiente ascendente muy pronunciada al inicio en *Precision* que coincide con el mejor umbral (que no es mejor que la pendiente de las gráficas de *EFFICIENTNET cam_pro* pero en la compensación *Recall-Precision*, *VGG* es mejor), el mejor modelo es *VGG cam_pro*.
2. **VGG cam:** el razonamiento es análogo al de *VGG cam_pro*, siendo peor que este porque su gráfica de *Recall* tiene una caída más rápida en su inicio.
3. **MOBILENET cam:** se vuelve a reutilizar el razonamiento de *VGG cam_pro*, donde es peor que los modelos de *VGG* porque su gráfica de *Precision* no tiene una pendiente ascendente tan larga y pronunciada como las de *VGG*.
4. **MOBILENET cam_pro:** este modelo está por delante de los *RESNET18* y *EFFICIENTNET* por su gráfica de *Precision*. Y está por detrás de *MOBILENET cam* porque, a pesar de tener aparentemente unas mejores gráficas de *Recall*, la pendiente ascendente de sus gráficas de *Precision* no son tan pronunciadas como la de *MOBILENET cam*.

5. **EFFICIENTNET cam_pro:** este modelo tiene unas malas gráficas de *Recall* que compensa con sus buenas gráficas de *Precision*, lo que hace que se posicione por delante de los modelos *RESNET*.
6. **RESNET18 cam_pro:** las gráficas de *Recall* tienen una mayor mejoría respecto de la de *RESNET18 cam* que la mejoría que tiene la gráfica de *Precision* de *RESNET18 cam* con respecto la de *RESNET18 cam_pro*
7. **RESNET18 cam:** por descarte tiene esta posición.
8. **EFFICIENTNET cam:** es el peor, su gráfica de *precision* dice claramente que no sabe localizar bien.

Como conclusión de esta comparación de modelos, el claro ganador es **VGG cam_pro**. Es el modelo que consigue alcanzar la marca más alta de entre todos los modelos con cualquier técnica. Además, fijando cada técnica de los modelos *cam_pro* (*Grad-CAM*, *Grad-CAM++* y *Smooth Grad-CAM++*), *VGG cam_pro* es el modelo que mejores marca saca en todas ellas. La explicación de este resultado viene de su gran capacidad de clasificar (atendiendo sobretodo a la métrica de *Recall* de clasificación), y del estudio de sus máscaras a lo largo de todos los umbrales sacando las métricas de *Iou*, *Recall* y *Precision* que nos dice que sus mapas de calor se centran en las regiones donde hay tejido cancerígeno con alta probabilidad. Todo esto arroja el resultado referencia de *intersection over union*:

$$IoU_{VGG \ cam_pro(Smooth \ Grad-CAM++)} = 0,6093$$

Si se compara con el *IoU_{sup}* (31) se tiene que:

$$\frac{IoU_{VGG \ cam_pro(Smooth \ Grad-CAM++)}}{IoU_{sup}} = \frac{0,6093}{0,836} = 0,73$$

esto quiere decir que el modelo *VGG cam_pro* consigue una capacidad explicativa de un 73 % con respecto la pseudocota superior *IoU_{sup}* (31).

4.4.3.3 Comparación de técnicas de explicabilidad

Se van a estudiar ahora las técnicas de explicabilidad para concluir qué técnica es mejor. Dado que *Grad-CAM* generaliza a *CAM* para redes con más de una capa totalmente conectada, en este estudio se van a comparar únicamente las técnicas de *Grad-CAM*, *Grad-CAM++* y *Smooth Grad-CAM++*.

Tomando la referencia de nuevo del Cuadro 9. El estudio consistirá en fijado un modelo, ver cómo se comportan las técnicas de explicabilidad con ese modelo.

En general, *Grad-CAM* y *Grad-CAM++* tienen un desempeño prácticamente idéntico en todas sus métricas dentro de cada modelo. Únicamente en *VGG*, *Grad-CAM* consigue ser superior. Por otra parte, *Smooth Grad-CAM++* en modelos como *RESNET18* y *MOBILENET* empeora bastante los resultados. Sin embargo, para *VGG* y *EFFICIENTNET* se comporta de manera similar a las otras dos técnicas.

Observando las gráficas solo se puede detectar algún tipo de diferencia entre *Smooth Grad-CAM++* con las otras dos. En la gráfica de *IoU* se ve como *Smooth Grad-CAM++* intenta aplanar la curva, siguiendo el criterio **C1**, esto significa que este tipo de técnica permite diferenciar mejor lo que el modelo entiende que es tejido cancerígeno y lo que no. Sin embargo, la altura de la en estas gráficas de *IoU* de *Smooth Grad-CAM++* es menor, lo que implica que le ha restado importancia a elementos clave, es decir, le ha bajado relevancia a partes de tejido cancerígeno dentro de las imágenes. Esto se ve desglosado mejor con la gráfica de *Precision* donde se pierde pendiente en *Smooth Grad-CAM++* con respecto al resto de gráficas en todos los modelos.

Además, hay que añadir que el coste computacional de *Smooth Grad-CAM++* es muy alto comparado con sus competidores. Como se fijó $N = 10$ (el número de imágenes con ruido), esto significa que el modelo tardará aproximadamente 10 veces más que *Grad-CAM++*. Sin embargo, entre *Grad-CAM* y *Grad-CAM++* apenas hay diferencia de tiempos, aunque *Grad-CAM* requiere menos cálculos.

Con todo esto se puede decir que a pesar de que en algunos modelos no sea la mejor técnica, *Grad-CAM* se puede decir que es, en balance, la mejor técnica, ya que consigue unos resultados similares o mejores a su competidor *Grad-CAM++* y aunque para el mejor modelo *Smooth Grad-CAM++* consiga un mejor desempeño, el costo computacional es lo suficientemente alto como para que la diferencia sea insignificante.

4.4.3.4 Mejor modelo y mejor técnica de explicabilidad

En esta última sección de análisis se va a recopilar la información del mejor modelo sobre todas las técnicas y de la mejor técnica sobre todos los modelos. Los resultados han dicho que el mejor modelo es **VGG cam_pro** y la mejor técnica es **Grad-CAM**. Así, se analizarán los resultados de esta técnica aplicada a este modelo.

Cogiendo el mejor modelo, **VGG cam_pro** estudiado, con la técnica que mejor se ha comportado para todos los modelos, **Grad-CAM**. Comparando su resultado de *IoU* el *IoU_{sup}* (31) se tiene que:

$$\frac{IoU_{VGG\ cam_pro(Grad-CAM)}}{IoU_{sup}} = \frac{0,6054}{0,836} = 0,724$$

esto quiere decir que el modelo *VGG cam_pro* consigue una capacidad explicativa de un 72.4 % con respecto la pseudocota superior IoU_{sup} (31). Un resumen de las estadísticas de clasificación se puede encontrar en el Cuadro 10, el de las estadísticas de localización se puede ver en el Cuadro 11 y en la Figura 34.

	Accuracy	Recall	Precision	F1-Score
VGG cam_pro	0.9912	0.9867	0.9939	0.9903

Cuadro 10: Resumen de estadísticas de clasificación de **VGG cam_pro** con la técnica de **Grad-CAM (GC)**. Extracto del Cuadro 8, donde este modelo conseguía los mejores resultados de *Accuracy*, *Recall* y de *F1-Score*.

Consigue unas estadísticas muy buenas de clasificación, concretamente en el *Recall* de clasificación (Cuadro 10) que es la métrica que mide lo bien que clasifica el modelo los casos patológicos que son justamente aquellos sobre los que se ha estudiado la localización.

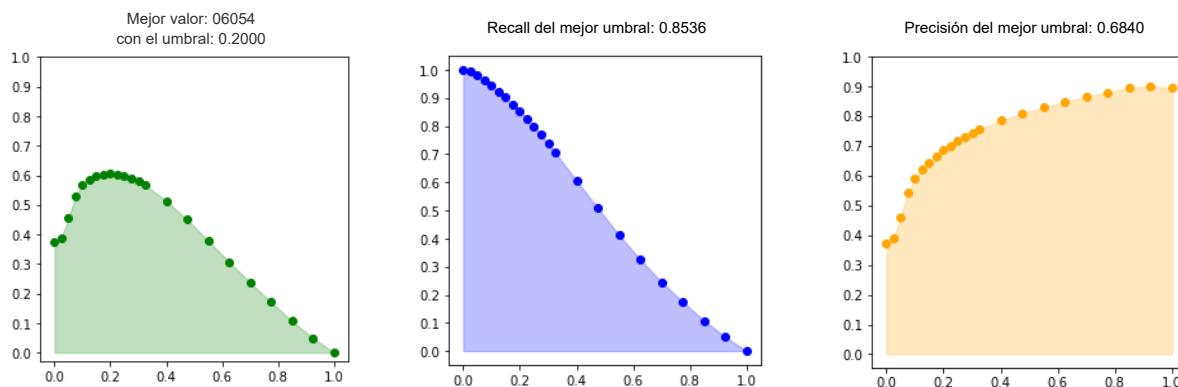


Figura 34: Gráficas de localización de **VGG cam_pro** con **Grad-CAM**.

	Umbral	IoU	IoU / IoU_{sup}	Recall	Precision	F1-Score
VGG cam_pro GC	0.2	0.6054	0.724	0.8536	0.684	0.7594

Cuadro 11: Resumen de estadísticas de localización de **VGG cam_pro** con la técnica de **Grad-CAM (GC)**. Extracto del Cuadro 9, donde a pesar de no conseguir en ninguna de las métricas los mejores resultados, mantiene un nivel muy alto, distanciándose muy poco de las mejores marcas.

En cuanto a las métricas de localización, a pesar de no ser el mejor resultado visto (*VGG cam_pro* con Smooth Grad-CAM++ conseguía una mejor marca), sus números son prácticamente similares al mejor.

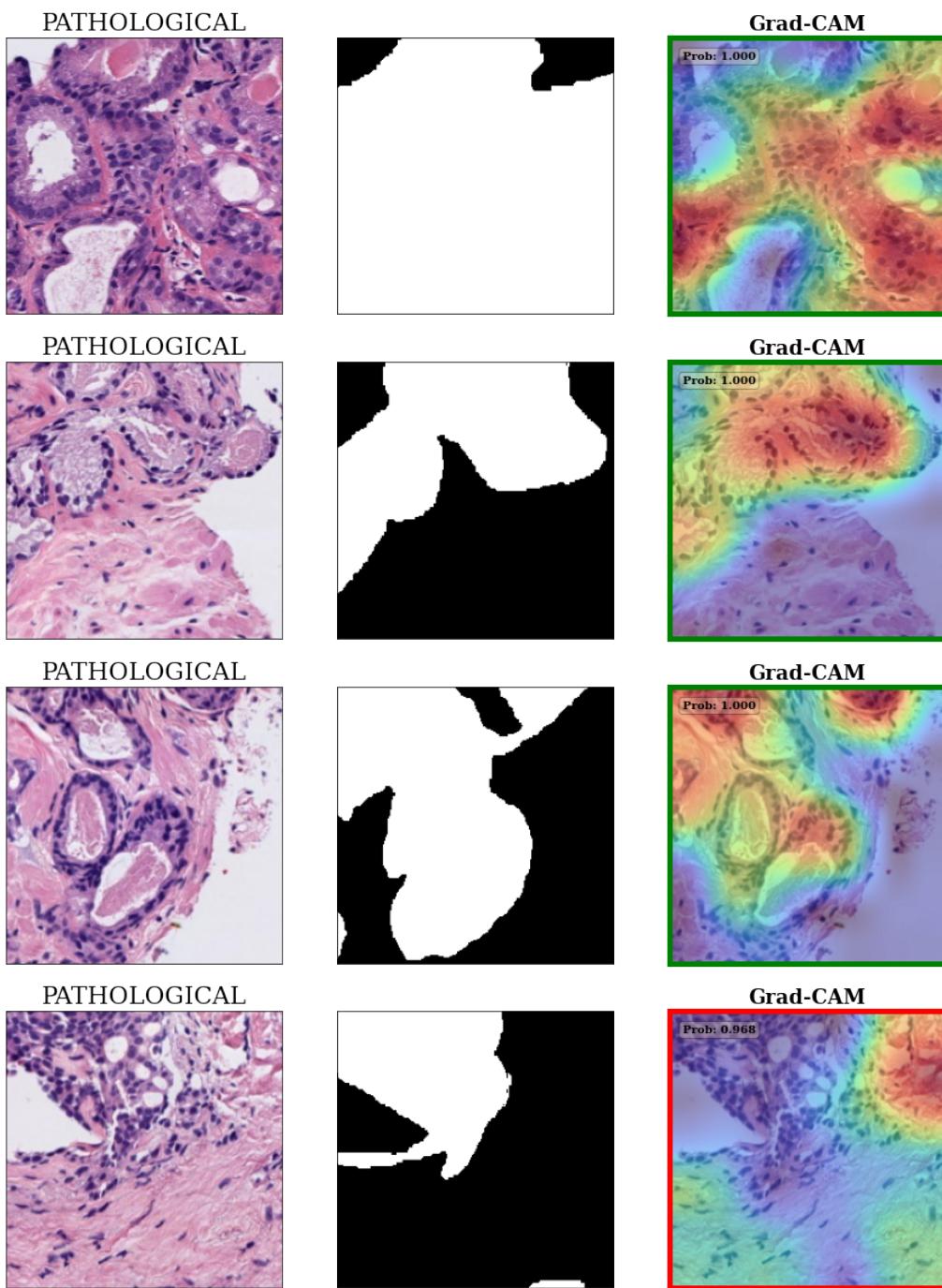


Figura 35: Ejemplo de mapas de calor dados por el modelo de VGG con la técnica de **Grad-CAM**. La máscara que aparece es la original. Comparando las máscaras con los mapas de calor se pueden sacar varias conclusiones. En el primer caso, *bien clasificado*, vemos que las regiones más claras no las selecciona probablemente porque el modelo lo confunde con fondo de la imagen. En el segundo y tercer caso, *también bien clasificados*, se centra muy bien en el tejido cancerígeno de la imagen. Por último, el caso de abajo es un ejemplo *mal clasificado*, viendo el mapa de calor se ve como el modelo se ha fijado en el tejido sano, lo que ha podido ser el foco de la confusión. De hecho, las probabilidades asignadas por el modelo son un 1. exacto para las que están bien clasificadas, pero sin embargo es algo menos para la imagen mal clasificadas, lo que significa que el modelo tiene, aunque sea, una ligera duda sobre la imagen.

4.5 CONCLUSIÓN

Tras la comparación de los modelos propuestos con las técnicas de explicabilidad de *CAM*, *Grad-CAM*, *Grad-CAM++* y *Smooth Grad-CAM++* se ha visto que la mejor técnica para este problema ha sido *Grad-CAM* ya que consigue unos valores de *intersection over union* que si no son los mejores para un modelo concreto, están muy cerca de serlo y con el menor coste computacional de las tres técnicas. La técnica de *Smooth Grad-CAM++* ofrece una dependencia fuerte del modelo y del conjunto de datos, lo que hace que no sea una técnica apropiada tanto por su tiempo de cómputo como por la dificultad de parametrizarla bien.

A pesar de conseguir una localización decente se ha visto que el uso de estas técnicas para localizar no podrá superar en muchas ocasiones a modelos que hayan sido entrenados de manera supervisada para la localización, ya que la redimension del mapa de calor al tamaño de la imagen original acarrea un error de partida. Además, la generalización de un mismo umbral para las imágenes normalizadas provoca incoherencias: se puede dar el caso de que la intersección de dos que forman parte del mismo WSI y son contiguas marquen tejido cancerígeno regiones distintas. Esta discontinuidad en las máscaras se debe al normalizar las imágenes por su valor máximo. Esto hace que las métricas no sean todo lo buenas que serían si se cogiese el umbral óptimo para cada imagen.

Por la parte de los modelos se ha visto como un modelo consigue mejores métricas de localización cuanto mejor clasifica.

Concluyendo con los resultados obtenidos se puede decir que la red *VGG16* con la técnica de *Grad-CAM* tiene un desempeño lo suficientemente bueno como para pasar a una fase de prueba donde expertos puedan usarla como herramienta de ayuda.

4.6 TRABAJO FUTURO

Se ha visto como la localización no supervisada puede abrir un abanico de posibilidades dentro de la medicina, ayudando a los expertos al diagnóstico de enfermedades, en este caso en imágenes histológicas para la detección de cáncer de próstata.

A pesar de que se han obtenido buenos resultados, estos podrían haber sido mejores si no hubiese tantas imágenes dentro del dataset con máscara trivial (toda la imagen es tejido patológico). Conseguir un conjunto de datos *libres de sesgos* como este y con un número mayor de imágenes podrán dar lugar a mejores modelos.

El estudio no se ha centrado en la arquitectura de los modelos, los cuales también tienen elementos que podrían afectar a la localización: la profundidad del modelo, el tipo de convolución (padding, stride...), las dimensiones de la entrada, las dimensio-

nes de la salida del extractor de características... Quizás se podría llegar a conseguir unos patrones para conseguir las redes que funcionan de manera óptima con estas técnicas de explicabilidad.

Otra vía de trabajo futuro sería estudiar una mejor manera de umbralizar. Con la umbralización actual para conseguir las máscaras se obtienen incoherencia entre imágenes, marcando como tejido cancerígeno zonas que en la imagen contigua a ella quizás no se marque como tejido cancerígeno como se ha explicado en la conclusión. Si se quisiera conseguir un umbral óptimo para todas las imágenes quizás se debería de barajar otro método, como por ejemplo normalizar las imágenes a través de un mismo máximo común.

Tras el estudio del mejor modelo y de la mejor técnica se podría prestar como herramienta a expertos para que valoren la ayuda que esta ofrece. Esta herramienta podría ser usada tanto en el diagnóstico de un paciente como para la docencia, ayudando a los alumnos a entender las imágenes sin ayuda del profesor.

BIBLIOGRAFÍA

- [1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Benetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, 2019. URL <https://arxiv.org/abs/1910.10045>.
- [2] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016. URL <http://cnnlocalization.csail.mit.edu/>.
- [3] 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. *International Journal of Computer Vision*, 128(2):336–359, oct 2019. doi: 10.1007/s11263-019-01228-7. URL <https://doi.org/10.1007%2Fs11263-019-01228-7>.
- [4] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, mar 2018. doi: 10.1109/wacv.2018.00097. URL <https://doi.org/10.1109%2Fwacv.2018.00097>.
- [5] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models, 2019. URL <https://arxiv.org/abs/1908.01224>.
- [6] Luis Merino and Evangelina Santos. *Álgebra Lineal con métodos elementales*. Paraninfo, Madrid, ESPAÑA, 2006.
- [7] C. Aparicio Del Prado R. Paya Albert. *Análisis Matemático 1*. Editorial Universidad de Granada, Granada, ESPAÑA, 1999.
- [8] Robert B. Ash. *Basic probability theory [by] Robert B. Ash*. Wiley New York, 1970. ISBN 0471034509.
- [9] Y.S. Abu-Mostafa, M. Magdon-Ismail, and H.T. Lin. *Learning from Data: A Short Course*. AMLBook.com, 2012. ISBN 9781600490064. URL <http://amlbook.com>, <http://work.caltech.edu/textbook.html>, /bib/abu-mostafa/abu2012learning/Yaser%20S.%20Abu-Mostafa%2C%20Malik%

20Magdon-Ismail%2C%20Hsuan-Tien%20Lin-Learning%20From%20Data_%20A%20short%20course-AMLBook.com%20%282012%29.pdf.

- [10] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. <http://www.deeplearningbook.org>.
- [11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL <https://arxiv.org/abs/1409.1556>.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL <https://arxiv.org/abs/1512.03385>.
- [13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. URL <https://arxiv.org/abs/1704.04861>.
- [14] Mingxing Tan, Le, and Quoc V. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019. URL <https://arxiv.org/abs/1905.11946>.
- [15] Wikipedia. René descartes — wikipedia, la enciclopedia libre, 2022. URL https://es.wikipedia.org/w/index.php?title=Ren%C3%A9_Descartes&oldid=142204189. [Internet; descargado 21-mayo-2022].
- [16] Wikipedia. Murray gell-mann — wikipedia, la enciclopedia libre, 2022. URL https://es.wikipedia.org/w/index.php?title=Murray_Gell-Mann&oldid=142914747. [Internet; descargado 21-mayo-2022].
- [17] Amirhosein Toosi, Andrea Bottino, Babak Saboury, Eliot Siegel, and Arman Rahimim. A brief history of ai: how to prevent another winter (a critical review), 2021. URL <https://arxiv.org/abs/2109.01517>.
- [18] A. M. Turing. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: [10.1093/mind/LIX.236.433](https://doi.org/10.1093/mind/LIX.236.433). URL <https://doi.org/10.1093/mind/LIX.236.433>.
- [19] By: IBM Cloud Education. What is machine learning?, 2020. URL <https://www.ibm.com/cloud/learn/machine-learning>.
- [20] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, apr 2021. doi: [10.1007/s12525-021-00475-2](https://doi.org/10.1007/s12525-021-00475-2). URL <https://doi.org/10.1007/s12525-021-00475-2>.
- [21] Machine learning: Qué es, tipos, ejemplos y cómo implementarlo, Feb 2021. URL <https://www.grapheverywhere.com/machine-learning-que-es-tipos-ejemplos-y-como-implementarlo/>.

- [22] Stanford University. Cs231n: Deep learning for computer vision stanford - spring 2022, 2022. URL <https://cs231n.github.io/convolutional-networks/>. [Online; accessed 8-May-2022].
- [23] Object detection guide, 2021. URL <https://www.fritz.ai/object-detection/>. [Online; accessed 11-May-2022].
- [24] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns, 2014. URL <https://arxiv.org/abs/1412.6856>.
- [25] Loris Bazzani, Alessandro Bergamo, Dragomir Anguelov, and Lorenzo Torresani. Self-taught object localization with deep networks, 2014. URL <https://arxiv.org/abs/1409.3964>.
- [26] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise, 2017. URL <https://arxiv.org/abs/1706.03825>.
- [27] Miguel López-Pérez Adrián Colomer María A. Sales Rafael Molina Ángel E. Esteban and Valery Naranjo. A new optical density granulometry-based descriptor for the classification of prostate histological images using shallow and deep gaussian processes. *Computer Methods and Programs in Biomedicine*, 178:303–317, 2019. ISSN 0169-2607. URL <https://www.sciencedirect.com/science/article/pii/S0169260719303906?via%3Dhub>.
- [28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- [29] Keras. Models stats on imagenet, 2022. URL <https://keras.io/api/applications/>. [Online; accessed 4-Jun-2022].
- [30] Siddharth Gupta, Salim Ullah, Kapil Ahuja, Aruna Tiwari, and Akash Kumar. Align: A highly accurate adaptive layerwise log_2_lead quantization of pre-trained neural networks. *IEEE Access*, PP:1–1, 06 2020. doi: 10.1109/ACCESS.2020.3005286.

A

APÉNDICE A. EJEMPLOS SOBRE EL CONJUNTO DE TEST

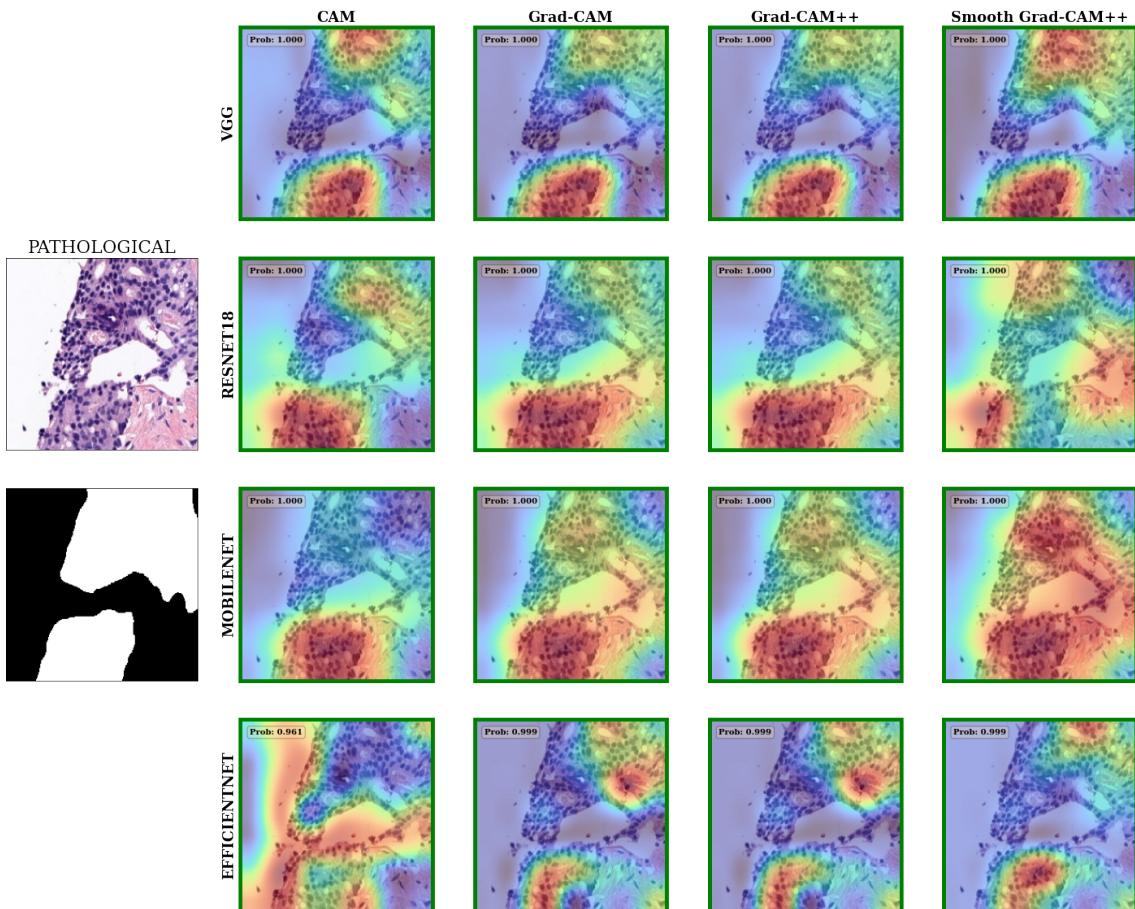


Figura 36: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. En este ejemplo todos los modelos exceptuando *EFFICIENTNET CAM* tienen un buen desempeño. Además se ve como la técnica de *Smooth Grad-CAM* amplía la zona de importancia, mejorando la calidad de la solución en algunos modelos como los de *VGG* y empeorándolo en otros como en *RESNET*.

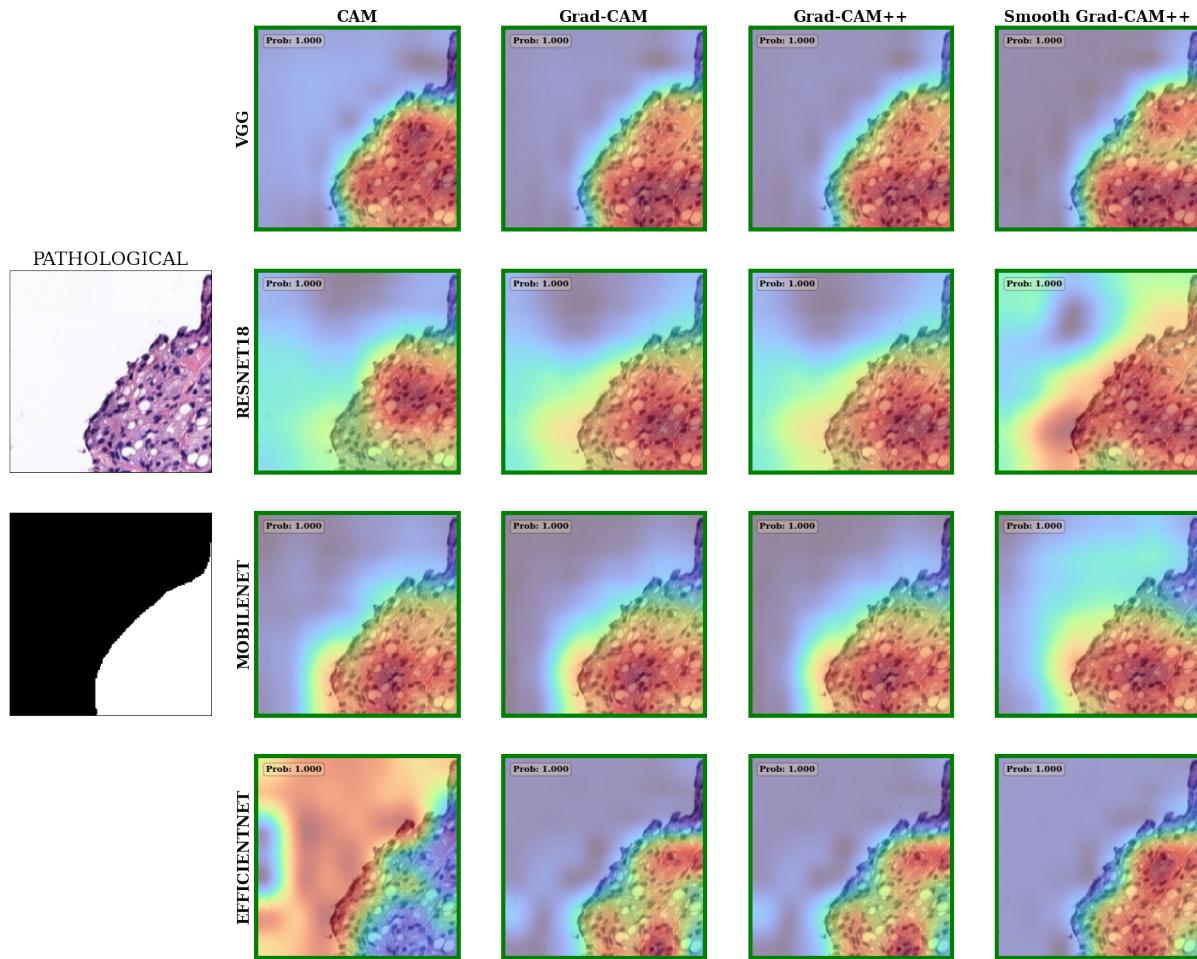


Figura 37: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. Este es un ejemplo donde la muestra de tejido es completamente cancerígena, el reto aquí es seleccionar bien el tejido completo. *EFFICIENTNET cam* recoge justo el fondo de la imagen, esto da una idea de lo mal que ha entrenado el modelo de *EFFICIENTNET cam* y da información para poner medidas para arreglarlo. El resto de modelos hace bien la detección siendo *VGG* el que tiene unos mejores modelos. *EFFICIENTNET cam_pro* también segmenta bien lo que es fondo de lo que es tejido.

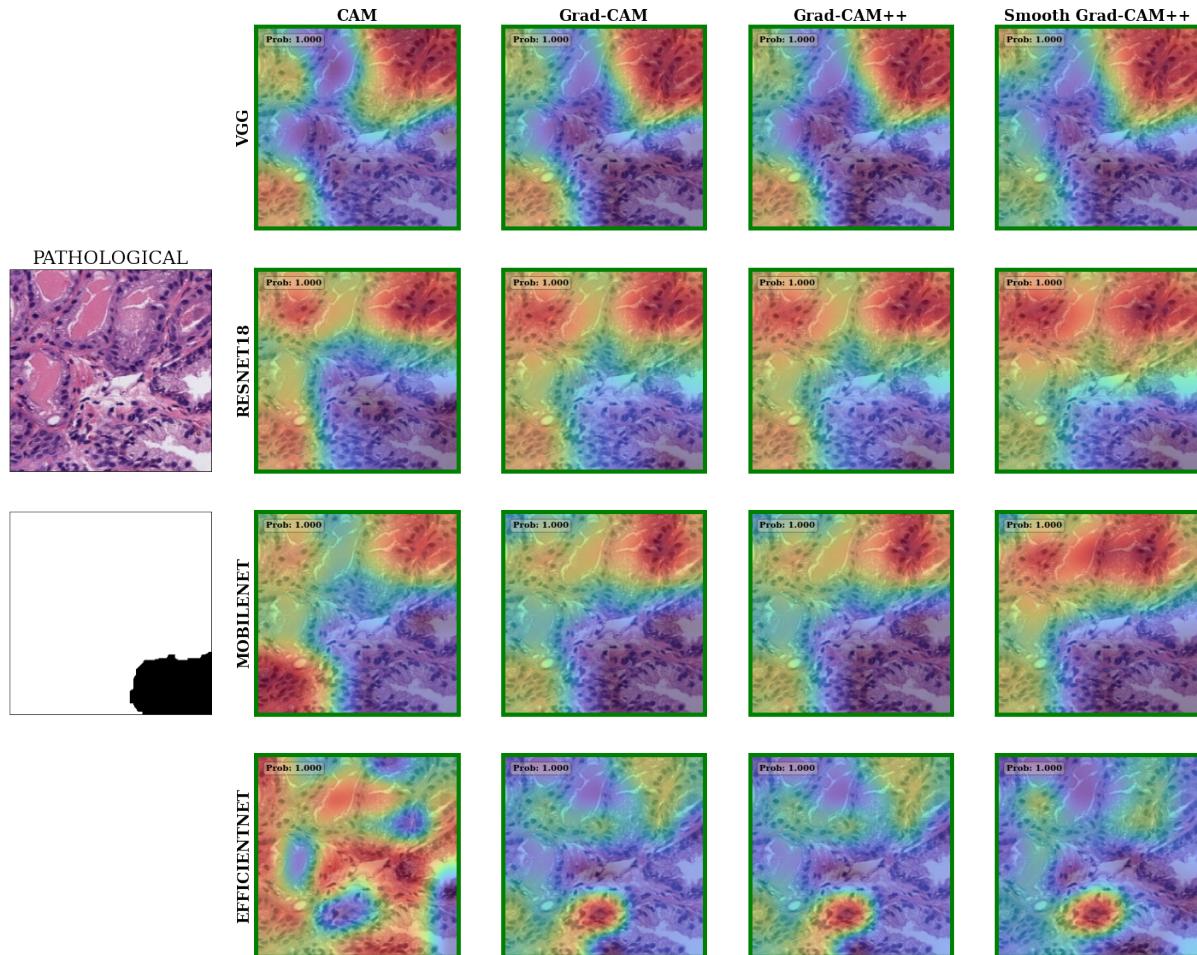


Figura 38: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. La importancia de este ejemplo es la cantidad de tejido cancerígeno que hay en la imagen. El reto es activar gran parte de la imagen. Ningún modelo logra activar toda la región necesaria, pero concretamente los de *EFFICIENTNET* tienen muy mal desempeño: el modelo *cam* por su mala localización nuevamente y el modelo *cam_pro* por la poca región que activa.

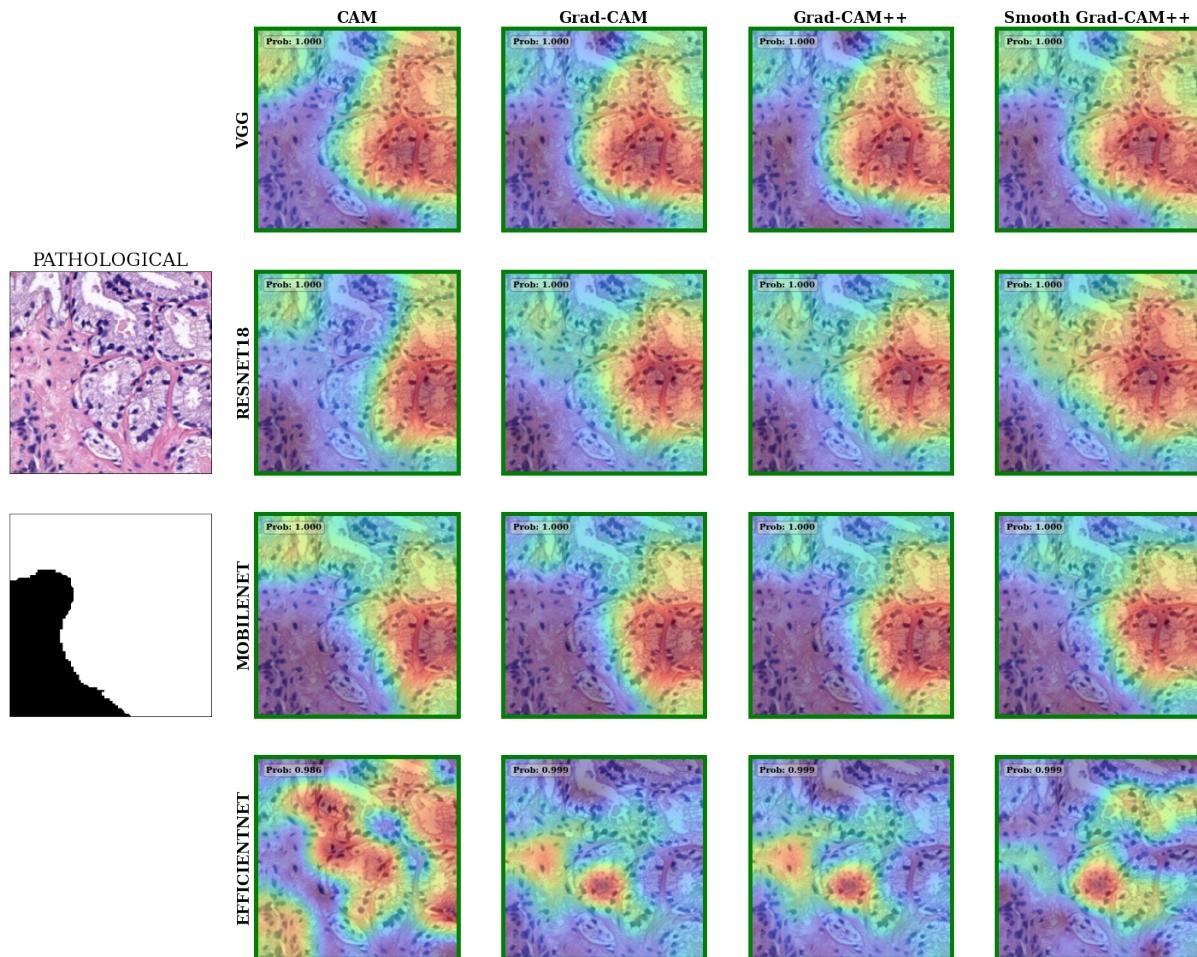


Figura 39: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. Ejemplo del que se pueden sacar las mismas conclusiones que la Figura 38

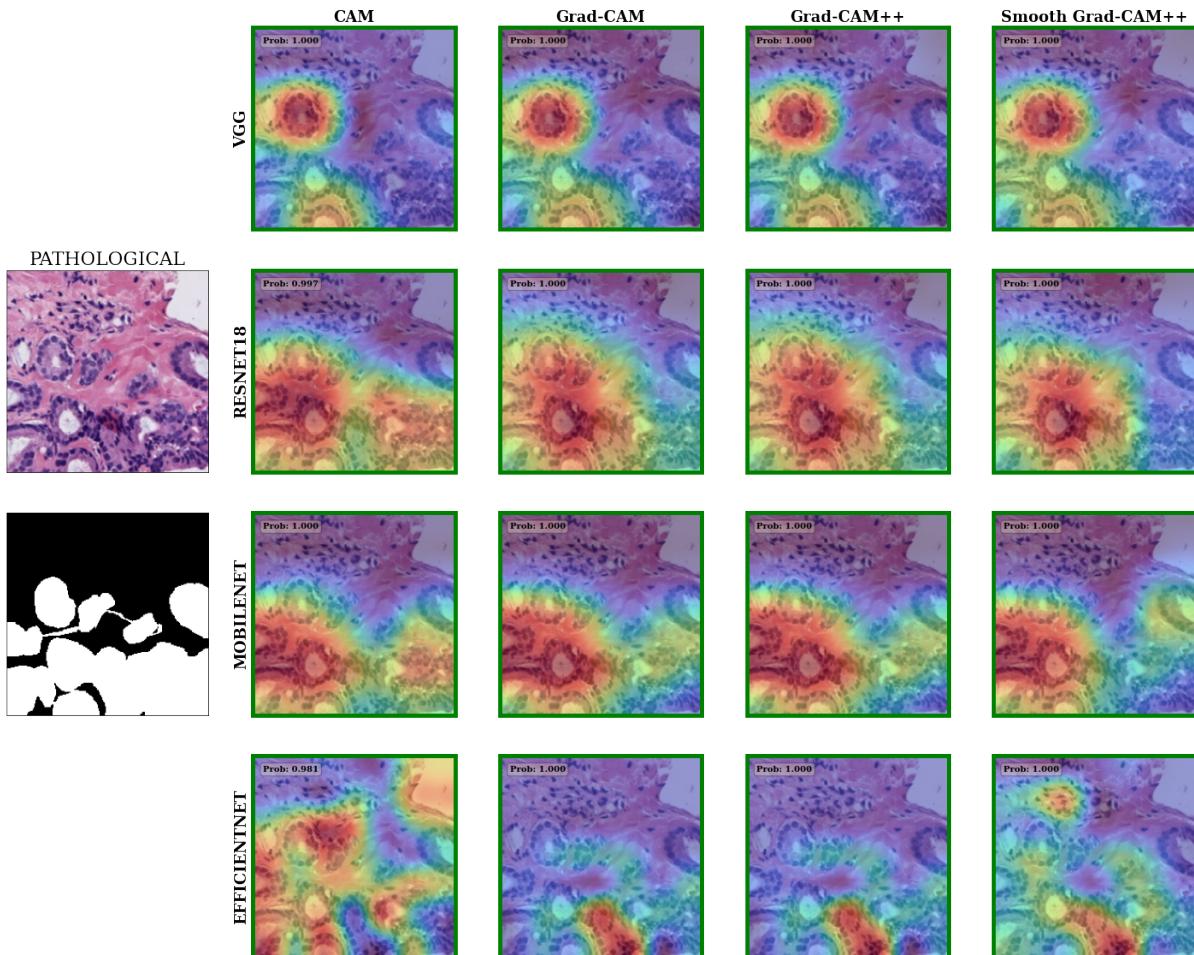


Figura 40: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. Este ejemplo pone en notoriedad como la redimensión de la salida del extractor de características hace imposible ajustarse al nivel de detalle de esta máscara, que contiene regiones cancerígenas casi independientes separadas por regiones muy pequeñas (se necesita mucho detalle) de tejido no cancerígeno.

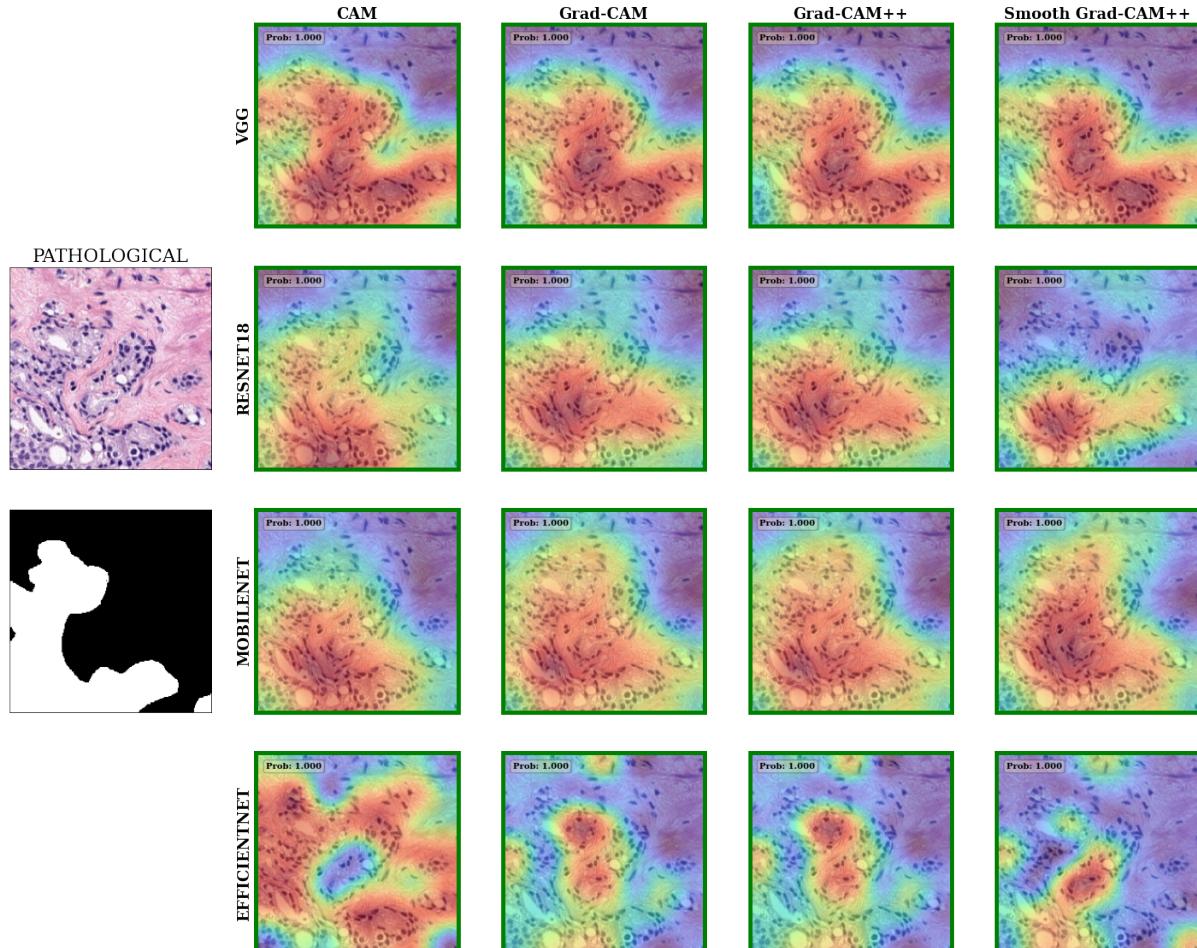


Figura 41: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. En este caso se ve como, menos con *EFFICIENTNET* el área activada por los modelos es más amplia que la del tejido cancerígeno. Esto sería un ejemplo donde el Recall es alto pero la métrica de Precision es baja.

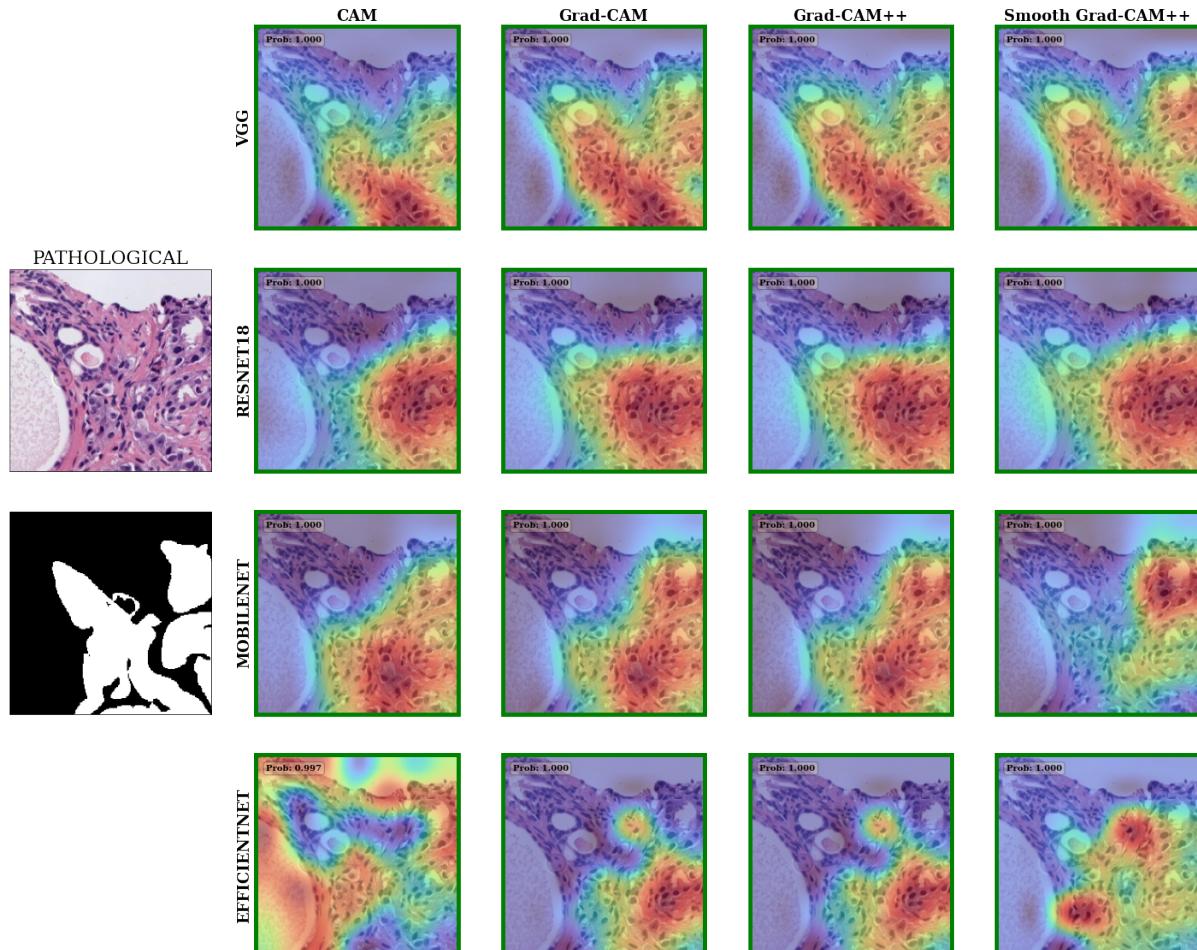


Figura 42: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. En este ejemplo también se ve afectada la precisión de las soluciones por el reescalado del mapa de calor. *EFFICIENTNET cam* vuelve a ser un mal modelo. El resto se centran mucho en la esquina inferior derecha, siendo los de *VGG* los mejores modelos.

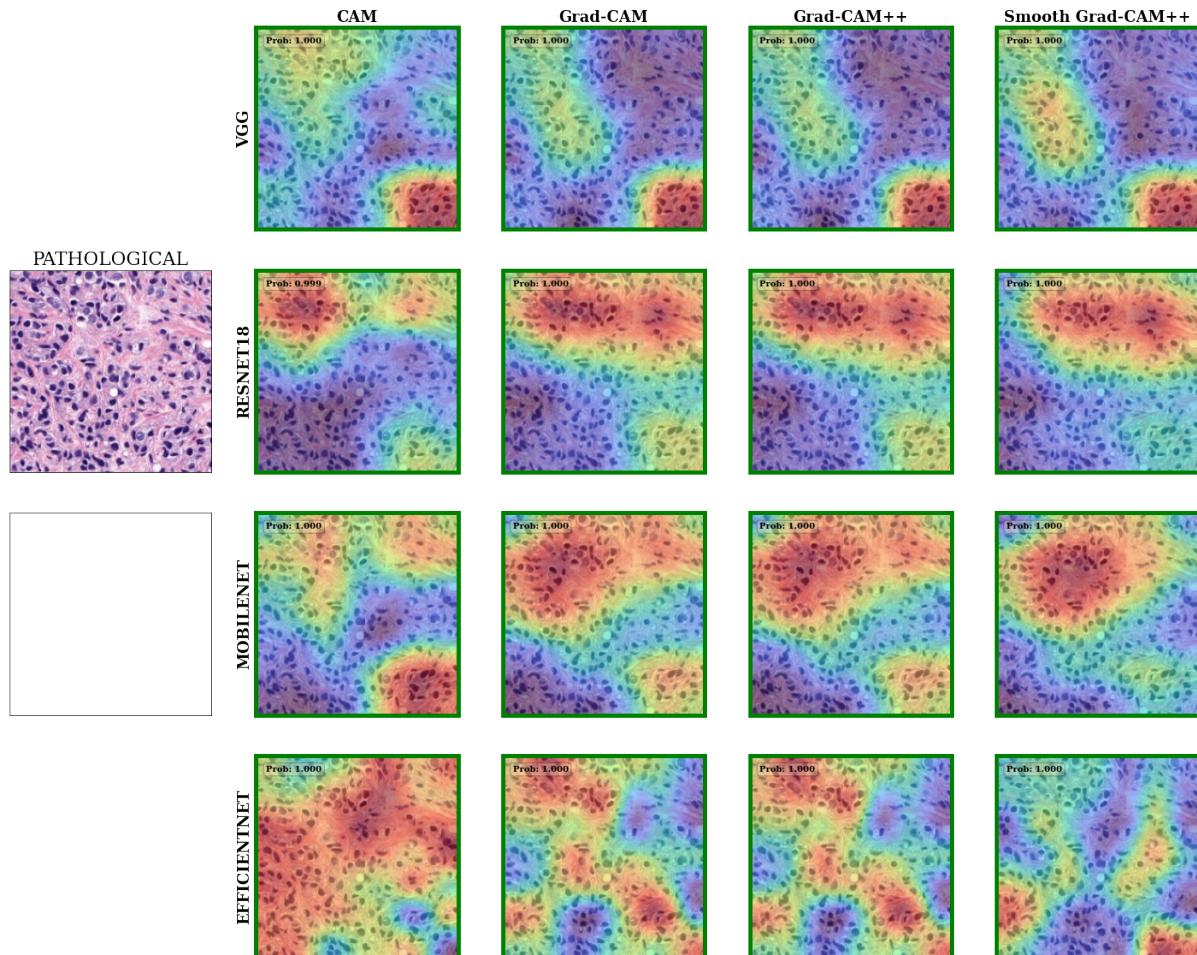


Figura 43: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. La importancia de este ejemplo es la cantidad de tejido cancerígeno que hay en la imagen. El reto es activar gran parte de la imagen. Aquí *EFFICIENTNET cam* es el mejor modelo, ya que es el que activa de manera más uniforme toda la imagen. Lo destacable de estos resultados es como, el que ha salido como peor modelo, da el mejor resultado en este ejemplo.

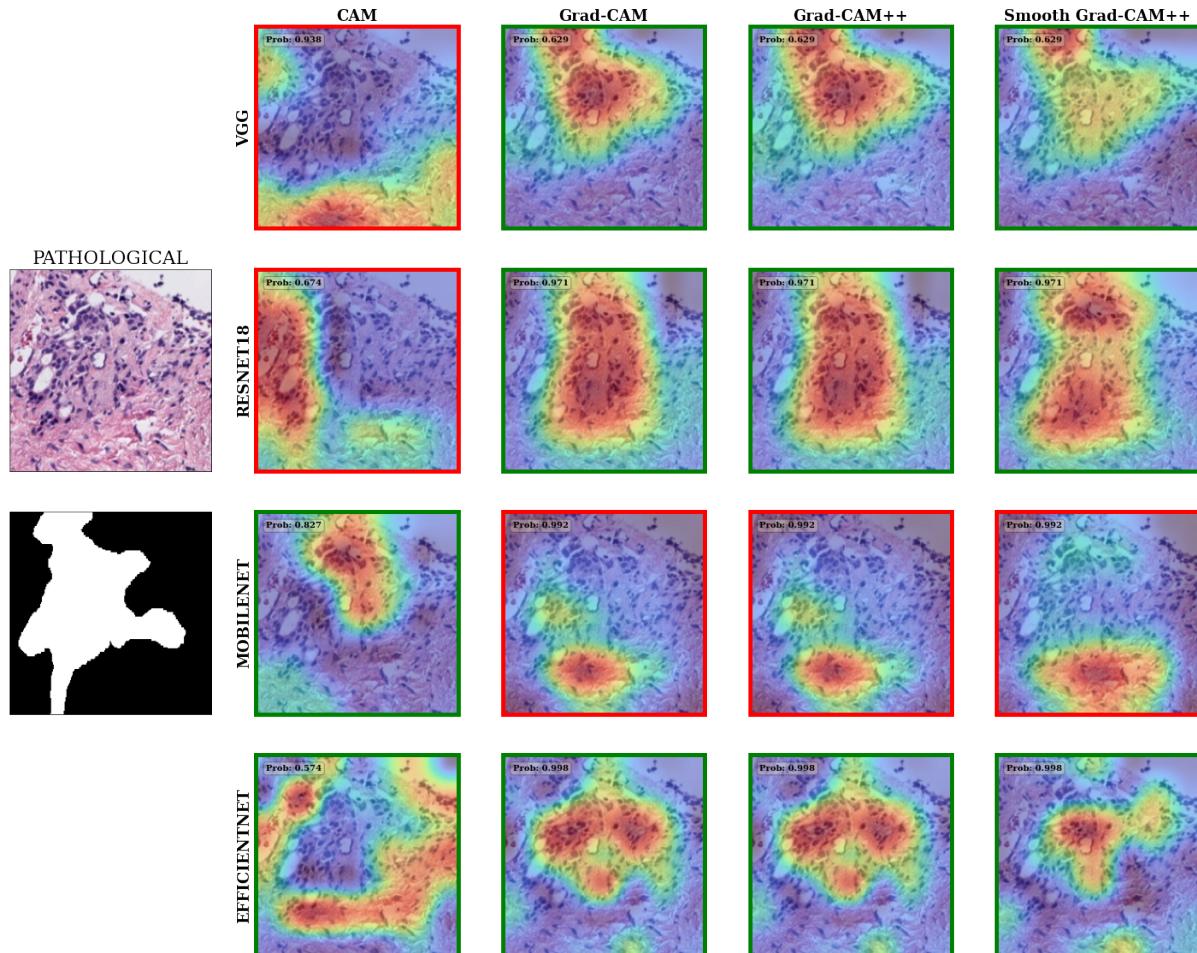


Figura 44: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. Ejemplo para mostrar debilidades de los modelos que fallan centrándose en el tejido benigno, algo que puede llegar a ser comprensible. De hecho se ve que el ejemplo es complejo porque ningun modelo da el 1.0 de probabilidad de la clase, resaltando la duda de la clasificación.

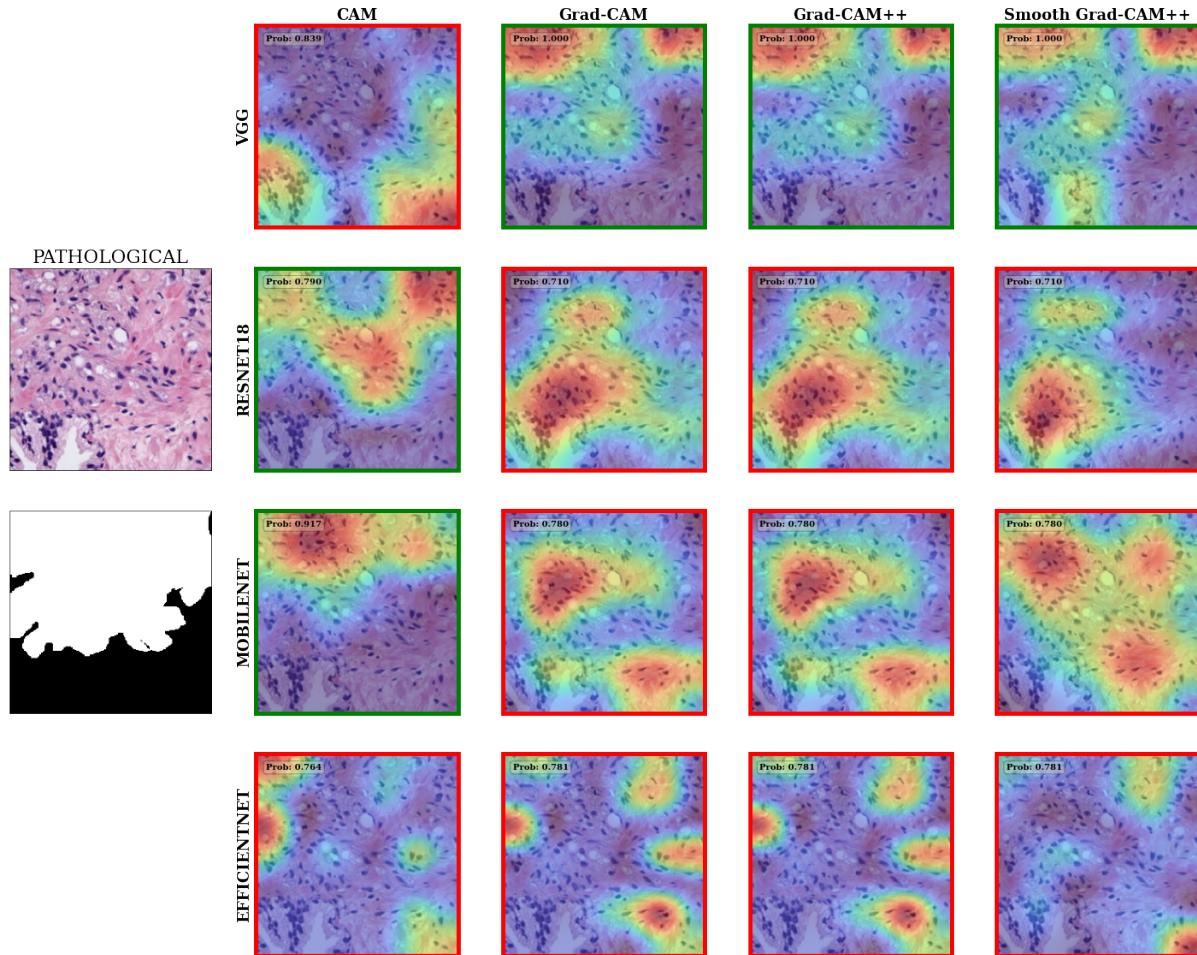


Figura 45: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. En este ejemplo muchas redes clasifican mal fijándose en el tejido que precisamente es cancerígeno. Esto significa que las características que dan relevancia a una clase no son coherentes.

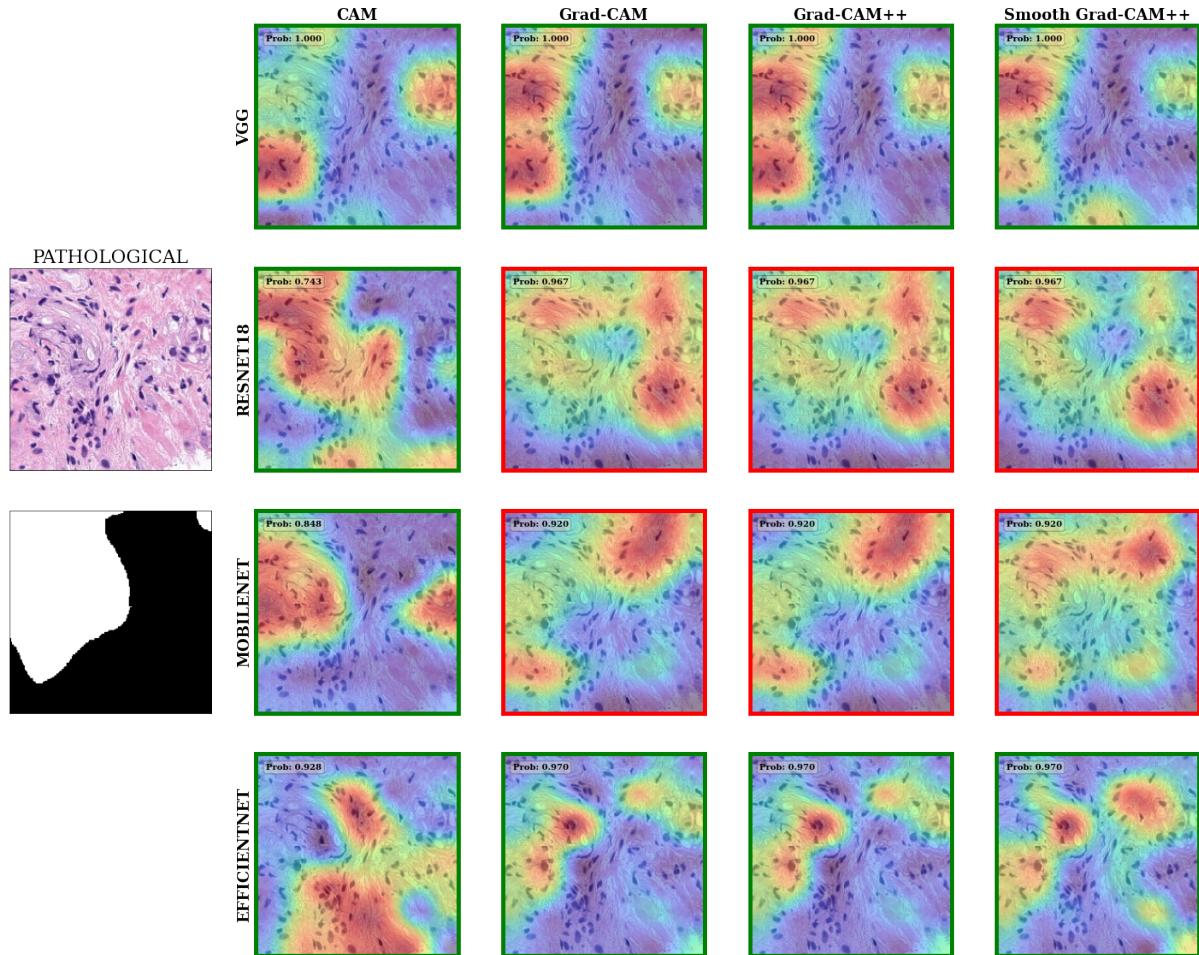


Figura 46: Ejemplo de imagen de test procesada por los modelos. En verde las imágenes bien clasificadas y en rojo las mal clasificadas. En cualquier caso el mapa de calor mostrado es el correspondiente a la clase patológica y la probabilidad mostrada es la probabilidad asociada a la clase predicha. Los ejemplos mal clasificados recogen tanto tejido cancerígeno como no cancerígeno, dejando ver como no segmenta bien las dos clases de tejido en este ejemplo.