Controlando el sesgo con técnicas de explicabilidad

Matías Aiskovich

UNIVERSIDAD AUSTRAL

Trabajo Final de Maestría en Explotación de Datos y Gestión del Conocimiento

Director: Andrés Araneo

Fecha: 18 de abril de 2023

UNIVERSIDAD AUSTRAL | INGENIERÍA

Índice general

Resumen

Las redes neuronales artificales poseen diversas aplicaciones en el campo de visión por computadora, como puede ser en agricultura, vehículos autónomos, reconocimiento facial, imágenes médicas y manufactura. En los últimos años se presenciaron grandes avances en estos algoritmos (He et al., 2015; Krizhevsky et al., 2012; LeCun et al., 1998), los que continúan ampliando el uso de esta tecnología.

La motivación de este trabajo surge con la inclinación de modelos de redes neuronales a aprender sesgos presentes en el conjunto de datos de entrenamiento (Li y Vasconcelos, 2019). Buscaremos como contrarestar esta inclinación y aprender a predecir correctamente evitando este sesgo.

En términos prácticos, se utilizaron métodos existentes para obtener información sobre los motivos de predicción en un modelo de visión por computadora. Se redujo el sesgo penalizando las activaciones incorrectas. Con esto se logró obtener un modelo que brinda activaciones más enfocadas en los objetos a detectar (visto mediante algoritmos que deducen la causa de la activación), que puede ser más robusto y performante, aun cuando se le presenten imágenes en situaciones atípicas. Esto resultó en una mejora de la metrica utilizada en este trabajo de 2.38 puntos sobre el conjunto de datos de validacion, y 2.69 puntos de mejora en los datos de testeo.

Abstract

Artificial neural networks have various applications in the field of computer vision, such as in agriculture, autonomous vehicles, facial recognition, medical imaging, and manufacturing. In recent years, there have been significant advances in these algorithms (He et al., 2015; Krizhevsky et al., 2012; LeCun et al., 1998), which continue to expand the use of this technology.

The motivation for this work arises from the tendency of neural network models to learn biases present in the training data set (Li y Vasconcelos, 2019). We will seek to counteract this tendency and learn to predict correctly while avoiding this bias.

In practical terms, existing methods were used to obtain information on the reasons for prediction in a computer vision model. The bias was reduced by penalizing incorrect activations. This achieved a model that provides more focused activations on the objects to be detected (as seen through algorithms that deduce the cause of activation), which can be more robust and perform better even when presented with images in atypical situations. This resulted in an improvement of 2.38 points in the validation data set and a 2.69-point improvement in the testing data set, based on the metric used in this work.

Palabras clave: Inteligencia Artificial, Visión por computadora, Redes neuronales artificiales, Interpretabilidad, *Visual explanations, Fairness, machine learning*

Agradecimientos

A mis padres, Ana y Eduardo Aiskovich y a mi novia Samantha Lee, quienes estuvieron presente durante el desarrollo y escritura de todo el trabajo.

A los médicos Diego Barros, Francisco Klein y Mauro Orlando, quienes me ayudaron en tiempos difíciles.

A mis compañeros de clase Laura Nóbile, Franco Donati, Tomas Lanza y Fernando Arrieta, por acompañarme en este proceso.

A Andrés Araneo por su apoyo, por el tiempo dedicado, sus valiosas sugerencias y predisposición para la concreción del presente trabajo.

Al doctor Juan Alé y a todo el cuerpo docente de la Universidad Austral por los conocimientos transmitidos a lo largo de la carrera.

Capítulo 1

Introducción

1.1. Motivación e importancia del tema de estudio

Las redes neuronales (Bishop, 1998), en particular, las del tipo convolucional, han demostrado su utilidad en una variedad de aplicaciones relacionadas con datos no estructurados, incluyendo visión por computadora (Voulodimos et al., 2018). Mencionando algunos de los ejemplos más relevantes, en el año 1998, LeNet (LeCun et al., 1998), demostró la capacidad que tienen las redes convolucionales de reconocer caracteres manuscritos y generó un salto cualitativo, en comparación con los algoritmos existentes al momento. Con la aparición del conjunto de datos ImageNet (Deng et al., 2010), se presentó una comparativa de rendimiento más compleja que el reconocimiento de caracteres manuscritos, que consistió en la clasificación de objetos en más de 20.000 categorías, y fue adoptada como una métrica universal para validar los avances en visión por computadora. AlexNet (Krizhevsky et al., 2012) entrenó un modelo basado en una arquitectura con capas convolucionales, utilizando la función de activación ReLU (Nair y Hinton, 2010), e implementando efectivamente GPUs para aumentar la velocidad de entrenamiento, lo cual fue crucial para poder entrenar una red del tamaño propuesto en una cantidad de tiempo razonable. Con esta arquitectura, ganó la competencia ILSVRC-2012 en el año 2012, con un error top-5 de 15.3 % (más de 10.8 % más bajo que el segundo puesto de la competencia), basado en ImageNet, y estableció un nuevo estado del arte para modelos de visión por computadora. Con este salto tan importante

de *performance*, demostró las posibilidades de aprovechar la capacidad de las arquitecturas basadas en redes convolucionales y del poder de cómputo provisto por las GPUs. ResNet (He et al., 2015) propuso una arquitectura basada mayormente en interconectar los datos de entrada de cada capa con capas posteriores, con el fin sobreponerse al desvanecimiento de gradiente en modelos profundos (Feng, 2017). Este modelo ganó la competencia ILSVRC 2015, con un error top-5 de 4.49 % (1.32 % más bajo que el segundo puesto).

En las investigaciones mencionados previamente, se evaluó la *performance* de estos modelos de clasificación de imágenes en métricas que no tienen en cuenta la causa de la activación de los mismos (Ils, 2011). Al desconocer las causas de la activación, por ejemplo, no podemos saber si el modelo predice que el objeto en una foto es un barco, porque aprendió las características que hacen a un barco, o porque aprendió que los barcos se encuentran sobre el agua. Esto podría significar que el modelo tenga una dificultad mayor en reconocer al objeto fuera de contexto, en nuestro ejemplo, el barco, en caso de que se encuentre fuera del agua. Este trabajo propone generar un modelo de clasificación de imágenes que sea más resistente a aprender relaciones de un contexto común en las imágenes de entrenamiento y que no necesariamente hagan al objeto a detectar. De esta forma, se otorgaría como beneficio modelos de visión por computadora más confiables, es decir, menos afectados por un sesgo existente en el conjunto de datos de entrenamiento.

En el entrenamiento de un modelo supervisado, brindamos datos para aprender, pero no tenemos una intervención sobre cuál es la forma de aprender de los mismos. Esto puede causar que un modelo haga una predicción correcta, por un contexto que se repite en los datos, pero que no necesariamente implique que el modelo haya logrado aprender cómo identificar el objeto a predecir. El impacto concreto va a depender del área de aplicación del modelo, por ejemplo, Taori et al. (2020) y Tommasi et al. (2017) enumeran el impacto del sesgo en modelos de visión por computadora en áreas tales como diagnóstico médico, seguridad, servicios financieros. Como es explicado por Tommasi et al. (2017) y Torralba y Efros (2011), este contexto común a los datos disponibles se presenta en la mayoría de los

conjuntos de datos, especialmente en aquellos que fueron generados por un grupo diverso de usuarios, en la modalidad *Crowdsourced*. El tema principal planteado en este estudio es cómo entender y guiar las causas de las activaciones de un modelo de redes neuronales.

Para entender las causas de las activaciones de un modelo de redes neuronales, utilizaremos algoritmos de explicabilidad, donde podremos visualizar las activaciones de esta red. Con esta información, durante el entrenamiento de la red, penalizaremos las activaciones no enfocadas en los objetos a predecir y luego evaluaremos el modelo resultante.

1.2. Requerimientos y desafíos

Se buscará entrenar un modelo de clasificación de imágenes que se enfoque en el objeto en sí, y que sea menos propenso a basarse en el contexto presente en los datos de entrenamiento. Partiendo de un modelo que sea estado del arte para la tarea de clasificación de imágenes, queremos reducir el efecto del contexto mediante nuevas técnicas propuestas en este trabajo. Con el fin de determinar si el modelo resultante cumple con este objetivo, se necesitarán datos que no hayan sido parte del proceso de entrenamiento.

Un gran desafío para nuestro planteo es el utilizar clasificadores de imágenes que ya resultan en performances muy altas, por ejemplo, ResNet 50 (el modelo base utilizado en este trabajo) logra un top-5 accuracy del 92.1 % en ILSVRC2015 (He et al., 2015). Otro desafío es que las métricas que son utilizadas en la mayoría de los trabajos de clasificación de imágenes se enfocan en medir si se predijo la clase correcta sobre cada imagen, sin importar la causa de esta predicción. Un modelo que haya aprendido a predecir la clase "Automóvil" si un semáforo está presente en la imagen puede tener una performance muy alta si se utiliza la métrica precisión y si este sesgo está presente en la mayoría de las imágenes. Por esto mismo, buscaremos utilizar una métrica que sea más descriptiva de la causa de activación y analizaremos las predicciones manualmente, con el fin de hacer un análisis cualitativo de nuestra propuesta.

1.3. Problemas no resueltos

En una red neuronal profunda, dada la cantidad de parámetros, y a diferencia de otros modelos de inteligencia artificial más simples, es muy difícil entender las causas de las activaciones. Con el fin de sobreponerse a esta problemática se han planteado distintos algoritmos de explicabilidad que buscan presentar al usuario información sobre lo que está causando la salida de la red neuronal. Es pertinente mencionar que los distintos algoritmos de explicabilidad, como Grad-CAM (Selvaraju et al., 2020), TCAV (Kim et al., 2017) y Score-CAM (Wang et al., 2019), llegan a la determinación de la causa de la activación de distintas formas y cada uno de estos algoritmos tiene distintas limitaciones, como es explicado por Tong y Kagal (2020) y Schaaf et al. (2021), estos metódos pueden ser más o menos eficientes en detectar el sesgo en las activaciones en una red. En este trabajo no planteamos la mejora de los algoritmos de explicabilidad, sino modelos que tengan un mejor rendimiento dentro de las limitaciones existentes en estos algoritmos.

Las funciones de pérdida y métricas comúnmente utilizadas para la tarea de clasificación de imagénes no suelen tener en cuenta la causa de las activaciones. Con el fin de sobreponernos a esta limitación, definiremos una función de pérdida y una métrica acorde que nos permita cuantificar los resultados de este trabajo. No obstante, debemos mencionar que esta métrica también tiene sus limitaciones en cuanto a su posibilidad de guiarnos en el objetivo primario que deseamos conseguir con este trabajo, un modelo que se active por el objeto en sí, y no por su contexto.

1.4. Solución propuesta

Se propone un método para dirigir el entrenamiento de una red neuronal, buscando guiar sus activaciones a la localización de los objetos en la imagen. Esto es distinto a la mayoría de los métodos de entrenamiento existentes, que tienen como fin el obtener la clase correcta como salida, ignorando las causas de estas predicciones. Utilizaremos un modelo existente que sea considerado estado del arte para la tarea de clasificación de imágenes. Por sobre este modelo, desarrollaremos un método para penalizar las activaciones de la red que estén localizadas por fuera de la caja delimitadora que contiene al objeto en sí. Para lograr esto, necesitaremos: 1) saber dónde se encuentra el objeto 2) entender qué parte de la imagen es la que activa a la red neuronal y causa la predicción resultante 3) comparar la información obtenida en los pasos anteriores y penalizar donde corresponda. Para cumplir con el punto 1), utilizaremos un conjunto de datos que cuente con cajas delimitadoras (*bounding boxes*) alrededor del objeto a predecir. Con el fin de resolver el punto 2), emplearemos información de algoritmos tales como Grad-CAM, el cual detecta qué regiones de la imagen se activaron para una predicción dada. Con esta información, en el punto 3), implementaremos una función de pérdida que combine entropía cruzada (Sección 2.7) y una penalización a las activaciones fuera de las cajas delimitadoras.

Esta solución podría generar un modelo que sea más robusto y en el que las activaciones estén más directamente relacionadas a los objetos que estamos intentando clasificar y no a su contexto, produciendo un modelo menos sesgado y evitando los errores mucha veces identificados en los modelos que fueron entrenados con un conjunto de datos sesgado.

Se realizó un estudio comparativo, utilizando los métodos de entrenamiento propuestos en este trabajo y los métodos de entrenamiento normalmente utilizados en la tarea de clasificación de imágenes, para así poder evaluar los aportes del presente trabajo. Es importante comprobar si, aún con las limitaciones mencionadas en los algoritmos de interpretabilidad (Schaaf et al., 2021; Tong y Kagal, 2020), logramos un modelo más consistente para el objetivo previamente explicado.

1.5. Trabajos relacionados

En el trabajo de Zhang et al. (2018) se propone un método para descubrir qué relaciones fueron aprendidas por una red neuronal convolucional y, mediante el análisis de esas relaciones, poder diagnosticar si alguna de ellas es espuria.

Iosifidis y Ntoutsi (2018) abordan el problema del sesgo en modelos de clasificación de datos tabulares, mostrando ejemplos donde modelos sesgan la clasificación de los ingresos de un individuo según su género. En este caso, se proponen técnicas para aumentar los datos e intentar reducir el impacto de este sesgo en las clasificaciones resultantes.

Li y Vasconcelos (2019) mencionan que algoritmos de *machine learning* pueden lograr una *performance* alta aprendiendo relaciones espurias. Explican esto por el hecho de que muchos conjuntos de datos contienen un sesgo común a todas las imagénes y el aprender este contexto puede ayudar a obtener un mejor rendimiento. No obstante, el aprender este contexto no es aprender a reconocer los objetos esperados (y no va a permitir identificar a los objetos si el sesgo no está presente en una situación distinta). En este caso, se propone penalizar el aprendizaje que le resulta demasiado fácil al modelo, tratando que el mismo no aprenda con sobre confianza relaciones que no son robustas.

En el trabajo de Mehrabi et al. (2019) se hace una revisión de las causas del sesgo en distintos modelos de *machine learning*, lo cual puede ser útil para comprender qué factores afectan esta problematica.

Choi et al. (2019) identifica las limitaciones de un modelo de clasificación de imágenes, en un contexto de reconocimiento de escenas, en cuanto a poder reconocer acciones atípicas para el contexto. En este análisis se propone utilizar una función de pérdida personalizada que emplea contenido especialmente preparado para esta forma de entrenamiento, donde en la imagen se oculta la acción concreta. La idea de esto es que el modelo no pueda predecir la acción correcta sin esa evidencia, incentivando a que él mismo identifique la acción a clasificar por el acto concreto y no se confunda con las acciones que son las más típicas del contexto (ej: poder clasificar un discurso en una cancha de baseball).

Singh et al. (2020) identifica el problema del sesgo en los modelos de visión por computadora. Plantea el objetivo de reconocer una categoría en la ausencia de su contexto y utiliza el algoritmo CAM para reconocer qué factores dentro de la imagen causan la activación del modelo. Luego, usa estos resultados para ocultar el contexto dentro de la imagen.

Clark et al. (2020) explica que en muchos *datasets* están presentes distintas correlaciones entre clases y contextos que no favorecen a los modelos que utilizan estos datos. Se propone entrenar dos modelos de redes neuronales, uno con una cantidad limitada de parámetros (un modelo relativamente simple) y un modelo con una alta cantidad de parámetros, exponiendo que el aprendizaje del modelo simple con un entrenamiento rápido va a ser, en general, relaciones espurias y esa es información que puede ser utilizada para penalizar al modelo más complejo.

Nuriel et al. (2020) identifica que las capas de BatchNorm pueden causar que el modelo aprenda relaciones sesgadas presentes en distintos lotes de entrenamiento. Plantea un nuevo tipo de capa de normalización enfocado en resolver problemas de sesgo. Se comparó las activaciones del modelo con el algoritmo Grad-CAM (Selvaraju et al., 2020), entrenado por el método propuesto y el que utilizó BatchNorm, obteniendo resultados mucho más enfocados en los objetos a predecir en el caso de la función de normalización propuesta por este trabajo. Así también, consiguen mejores resultados en benchmarks públicos tales como CIFAR100 o ImageNet.

Tong y Kagal (2020) identifican la problemática del sesgo de género en modelos de clasificación de imágenes. Muestra casos donde un clasificador de imágenes predice la imagen de un hombre con instrumental sanitario como un médico, mientras que en el caso de una mujer con la misma indumentaria, el modelo predice enfermera. Aquí se analiza la posibilidad de implementar un algoritmo, como Grad-CAM, con el fin de encontrar sesgo en un modelo, y se manifiestan distintos métodos para medir y cuantificar este sesgo.

Schaaf et al. (2021) analizan distintos tipos de algoritmos de explicabilidad, tal como Grad-CAM. Explora la capacidad de estos en detectar un modelo sesgado, utilizando para

ello casos sesgados generados sintéticamente.

Pillai y Pirsiavash (2021) intentan guiar el entrenamiento del modelo con la meta de obtener resultados enfocados en los objetos a predecir cuando se utiliza el algoritmo Grad-CAM. Para esto, aplicó una función de pérdida que penaliza las activaciones de Grad-CAM, cuando estas no están dentro de la imagen que contiene al objeto. Con el fin de determinar dónde está el objeto, usó aprendizaje autosupervisado, creando ejemplos de entrenamiento combinando una imagen que contenía el objeto a predecir y tres otras imágenes que no contenían el objeto a predecir. Se penalizaron las activaciones que sucedieron fuera de la imagen positiva. Esta propuesta es muy interesante, pero tiene la limitación de no permitir penalizar los sesgos que comúnmente ocurren en todas las imágenes correspondientes a la misma clase en un conjunto de datos dado.

Otro caso con aprendizaje autosupervisado es el de Pillai et al. (2021), donde se penalizan activaciones en imágenes que no contienen la clase a predecir, y se aumenta el área donde ocurrió la activación de Grad-CAM, en la imagen que contiene la clase correcta. Esta información es utilizada por la función de pérdida propuesta por este trabajo. En este caso, las fuentes de datos utilizadas no permiten, por lo menos de forma directa, el penalizar sesgos causados por relaciones que aparecen frecuentemente en las imágenes, ya que dependen de la penalización de los objetos presentes en imágenes elegidas al azar.

1.6. Aporte original de este trabajo

Como hemos explicado en la sección anterior, podemos ver que muchos estudios anteriores reconocen el problema del sesgo en los modelos de inteligencia artificial. Asimismo, varios de estos exploraron distintas soluciones para limitar este sesgo. La principal diferencia de nuestra propuesta, es que guiamos las activaciones, basado en la salida de Grad-CAM, hacia el cuadro delimitador de donde se encuentra el objeto. Esto nos brinda una especial ventaja en los casos en donde el sesgo no sea específico a una imagen en particular, sino a un sesgo común al conjunto de datos. Por ejemplo, si todos los barcos presentes dentro de un conjunto de datos se encuentran sobre el agua, los trabajos discutidos que utilizan el aprendizaje autosupervisado no podrán penalizar las activaciones causadas por el agua, debido a que se basan en penalizar las activaciones que ocurren fuera de la imagen que contiene la clase correcta. Al momento de la publicación de este estudio, no conocemos de la existencia de otro trabajo que haya utilizado esta misma técnica.

Capítulo 2

Métodos

2.1. Redes neuronales artificiales

En el presente ensayo utilizamos un modelo de redes neuronales. El perceptrón (Rosenblatt, 1958) es el modelo matemático más simple de una red neuronal artificial, que fue propuesto en el año 1958 por Rosenblatt. Consiste en un modelo de clasificación binario, del tipo lineal, de una sola capa, que realiza una predicción combinando un conjunto de pesos con un vector de entrada. Por ser una red de una sola capa, no es posible aprender una relación no lineal con este tipo de modelo. Este obstáculo fue solucionado con la adición de múltiples capas, que resultó en el modelo conocido como *multi-layer perceptron*. En el año 1969, Minsky publico *Perceptrons: an introduction to computational geometry* (Minsky y Papert, 1969), donde explicaba las limitaciones de estos modelos y las limitaciones en el poder de cómputo existente. En el año 1973, Dreyfus propuso el método de *back-propagation*, que permitió un entrenamiento práctico de las redes multicapas. Los avances en los nuevos métodos de redes neuronales fueron acompañados por el incremento del poder de cómputo disponible año a año.

Las redes neuronales artificiales proveen un método eficiente para modelar datos complejos. Como es explicado por Abiodun et al. (2018), en muchos ámbitos, las redes neuronales obtienen resultados que son considerados estado del arte. En este trabajo, nos enfocamos en los modelos de redes neuronales artificiales que aprenden de pares de objetos: datos de

entrada (en este caso, imágenes) y de salida, los que son los resultados deseados. Este tipo de entrenamiento es también conocido como aprendizaje supervisado. Una red neuronal es un conjunto de neuronas, donde cada una de estas posee la capacidad de calcular una suma ponderada de sus entradas, y luego aplicarle una función de activación. El resultado de salida (luego de la activación) le es transmitido a la próxima neurona de la red. Las redes neuronales cuentan con una cantidad variable de capas, y dentro de cada capa, una cantidad variable de neuronas. Durante el entrenamiento, se busca minimizar una función de pérdida, y mediante este proceso se ajustan los parámetros de las neuronas y de las conexiones entre las neuronas de cada capa ponderada por los pesos. Aquí, nos vamos a enfocar en las redes del tipo *feedforward neural network* (FNN), las cuales no tienen ciclos o iteraciones dentro de la red y, en las cuales, la información fluye solo en una dirección, hacia adelante (*forward*), desde la entrada, pasando por las capas ocultas (si las hay) hasta la salida. Las funciones de activación que suelen implementarse son del tipo no lineal, ya que el usar una función lineal generaría que el modelo sea equivalente a una regresión lineal. Es en el poder modelar datos no lineales donde las redes neuronales suelen demostrar su capacidad (Bishop, 1998).

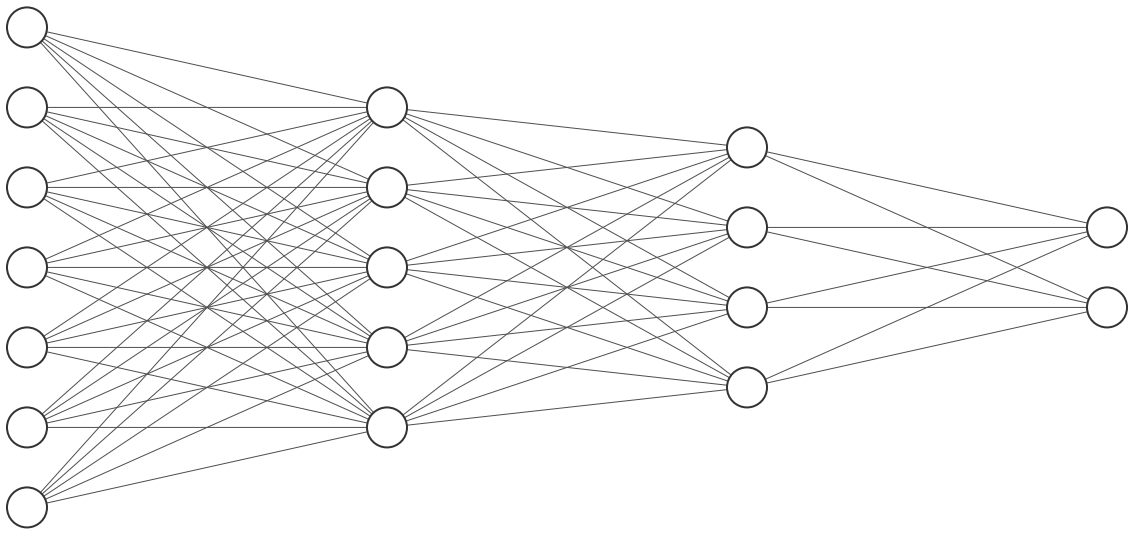


Figura 2.1: Red FNN

2.2. Redes neuronales convolucionales

Las redes neuronales *fully connected* tienen limitaciones para ser invariantes a la posición de los objetos a detectar (también conocido como *translation invariance*). Una red de este tipo aprende a reconocer patrones en ciertos sectores de las imágenes. Si ese patrón puede aparecer en otro sector de la imagen, este aprendizaje tiene que ser replicado en la neuronas que reciben como entrada este sector, incrementando fuertemente la cantidad de parámetros necesarios en la red.

Una red neuronal convolucional es un tipo de red en la que se cuenta, como mínimo, con una capa del tipo convolucional, donde no todas las neuronas de una capa están conectadas con las neurona de la próxima capa, y en la cual se utiliza la operación matemática conocida como convolución. La convolución permite obtener filtros invariantes al cambio, que luego son aplicados en muchos sectores de la imágen. Esto permite que un solo filtro reconozca un objeto cualquiera sea la posición del mismo. Como resultado, se obtienen redes más pequeñas que una red *fully connected* de una *performance* similar.

Las redes neuronales convolucionales se consideran históricamente basadas en las redes Neocognitron (Fukushima, 1980), que proponían una estructura jerárquica, compuesta de unidades capaces de reconocer patrones sin importar su posición en la imagen. El trabajo de Fukushima refiere su inspiración al trabajo de Hubel y Wiesel (Hubel y Wiesel, 1962) de explorar el funcionamiento de las neuronas de un gato e intentó replicar parte de este funcionamiento. El modelo Neocognitron introdujo las capas convolucionales y las capas de reducción de resolución. Otros avances significativos en la historia de las redes neuronales convolucionales fueron la adopción de las capas de Max Pooling (Yamaguchi et al., 1990), la red LeNet (LeCun et al., 1998) (donde se demostró la capacidad de las redes convolucionales en reconocer caracteres manuscritos), y la utilización de GPUs para acelerar el entrenamiento de los modelos (Oh y Jung, 2004).

Una convolución es un operador matemático que transforma dos funciones en una tercera función. Existen muchas convoluciones que se han utilizado en el procesamiento de imáge-

nes, tales como Gaussian Blur (Szeliski, 2022), Sharpen (Szeliski, 2022) o Edge Detection (Szeliski, 2022), donde a cada región de la imagen se le aplica una convolución con un filtro estático (los valores del filtro están fijos). En una red neuronal, el concepto de convolución es similar, pero estos filtros son aprendidos mediante datos de entrenamiento. Esto permite generar filtros que identifican ciertas características particulares de los datos con los que contamos. Por ejemplo, si lo utilizamos en imágenes con el fin de identificar vehículos en estas, uno de los filtros aprendidos podría ocuparse de detectar ruedas en las imágenes. Dentro de una capa convolucional, se cuenta con una cantidad a elegir de neuronas convolucionales. En cada neurona, se cuenta con filtros de convolución, que son aprendidos mediante los datos de entrenamiento. Estos filtros de convolución son aplicados sobre toda la imagen, permitiendo aprender patrones que son invariantes a la ubicación. Esto nos permite identificar, con el mismo filtro, un patrón aprendido, aun si el mismo aparece en una parte de la imagen distinta que en los datos de aprendizaje. El hecho de aplicar una misma convolución a distintos sectores, y el tener menos conexiones entre las neuronas, permite tener una cantidad menor de parámetros que una red equivalente del tipo *fully connected* (donde cada neurona tiene una conexión con todas las otras neuronas de la próxima capa). Esta ventaja en la cantidad de parámetros, permite manejar una cantidad mayor de datos de entrada (en el caso de las imágenes: su cantidad y resolución).

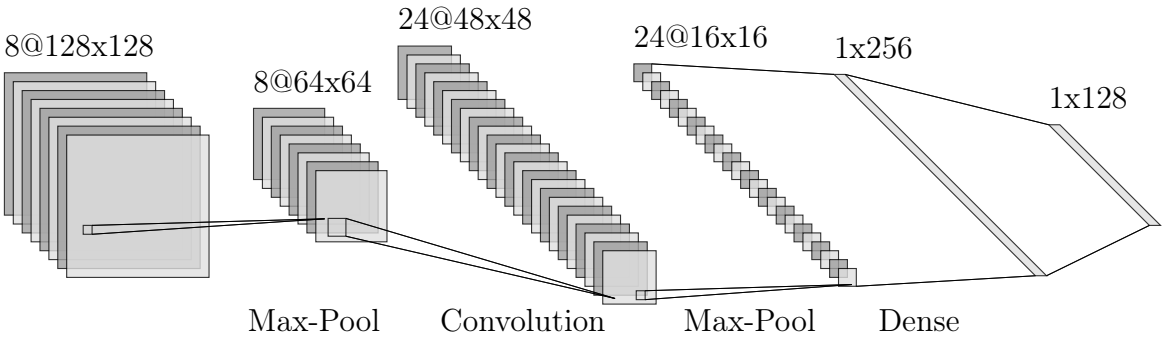


Figura 2.2: Red CNN

2.3. ResNet

Los modelos de redes neuronales con muchas capas, en general, presentan el problema de vanecimiento de gradiente (Feng, 2017), lo que hace muy difícil su entrenamiento. Hubo muchos intentos por sobreponerse a este problema. Szegedy et al. (2015) intentó añadir una función de pérdida auxiliar en las capas intermedias, pero esto no demostró ser una solución concluyente. El modelo ResNet fue presentado en el año 2015, y propuso una arquitectura basada principalmente en la idea de conexiones de acceso directo de identidad, donde la entrada de una capa se conecta directamente con capas más profundas de la red. Esto permite que la entrada se conecte directamente con capas alejadas, evitando la degradación de estas entradas y facilitando el entrenamiento de redes con muchas capas (He et al., 2015). En los experimentos presentados en la publicación original de ResNet, se pudo entrenar una red profunda con 1001 capas, lo que otorgó mejores resultados que una red menos profunda. Por estos resultados, ResNet se volvió una de las arquitecturas más populares en muchas tareas de visión por computadora (Feng, 2017). La arquitectura base de este análisis se basó en el modelo conocido como ResNet, en particular, en su version ResNet50.

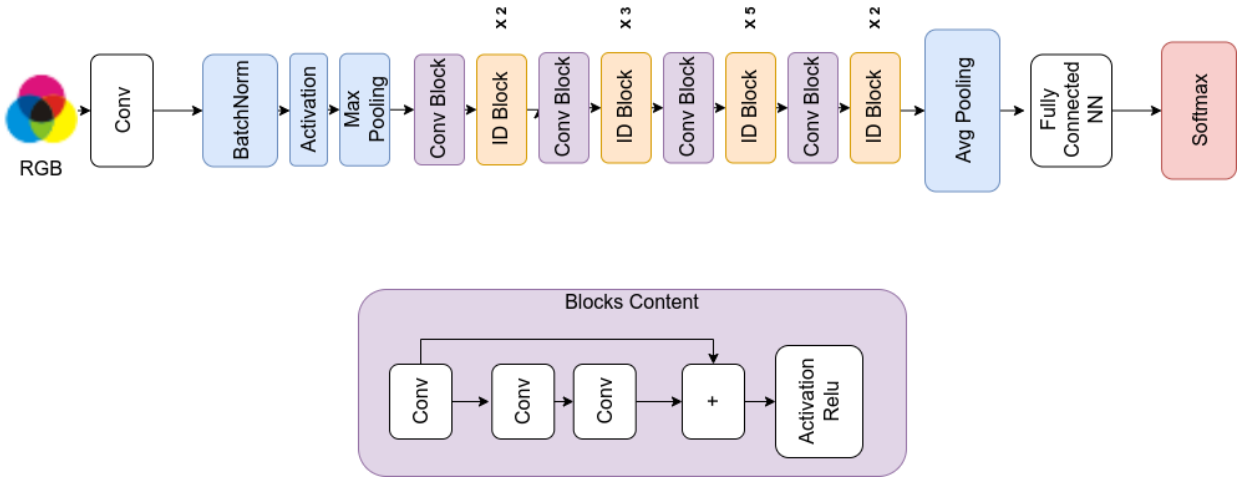


Figura 2.3: Red Resnet50

2.4. *Transfer Learning*

El término *transfer learning* (Weiss et al., 2016) se refiere a transferir el conocimiento aprendido por un modelo a una tarea distinta para la que fue entrenado originalmente. Esto consiste en tomar un modelo que ya fue entrenado para una tarea, con sus pesos modificados durante este entrenamiento, tarea en la que en general se cuenta con muchos datos de entrenamiento, y reentrenar este modelo, parcial o completamente, para la nueva tarea (en general, la cantidad de datos disponibles para la segunda tarea es mucho menor que para la primera). Aun cuando la nueva tarea guarda poca relación con la tarea original, se demostró que, en general, esto permite una ganancia significativa de *performance* (Ravishankar et al., 2016). Las librerías comúnmente utilizadas para el entrenamiento de redes neuronales, tal como PyTorch (Paszke et al., 2019), suelen contar con versiones preentrenadas de arquitecturas de modelos conocidos disponibles para su descarga. El poder de esta técnica radica en que brinda una potencial solución a los casos en los que no se cuenta con una cantidad suficiente de datos de entrenamiento, y/o con los recursos suficientes para entrenar un modelo sobre una gran cantidad de ejemplos. Esto permite obtener, por lo general, mejores modelos con recursos limitados. Esta técnica es implementada en diversas aplicaciones de los algoritmos de inteligencia artificial, y no está limitada a la visión por computadora. En el caso de procesamiento de lenguaje natural (NLP, por sus siglas en inglés), es utilizada en muchos de los modelos actuales (Ruder et al., 2019), donde un modelo aprende de una tarea autosupervisada sobre un corpus muy grande (Devlin et al., 2018) y luego es ajustado para una tarea particular con la que se cuenta con un conjunto de datos limitado.

2.5. Grad-CAM

Las redes neuronales profundas son muchas veces consideradas cajas negras (*black-box*) (Burkart y Huber, 2021). Se utiliza este término para modelos complejos, con millones de parámetros, como por ejemplo, AlexNet(Krizhevsky et al., 2012) con 62 millones de parámetros, donde se vuelve muy difícil entender qué causó que un modelo resulte en una predicción determinada. Demostrando este problema, Szegedy et al. (2013) creó ejemplos sintéticos: agregando un ruido imperceptible al ojo humano, se lograba confundir a un clasificador de imágenes, aun cuando pudo predecir correctamente la imagen original. Esto es conocido como *Adversarial Attack* (Szegedy et al., 2013).

Con el fin de intentar dar una explicación comprensible a las causas de activación de una red neuronal (y de distintos algoritmos de inteligencia artificial), surgieron diversas propuestas (Bany Muhammad y Yeasin, 2021; Burkart y Huber, 2021; Ibrahim y Shafiq, 2022; Selvaraju et al., 2020; Wang et al., 2019). *Gradient weighted Class Activation Mapping* (Selvaraju et al., 2020): es un método para producir "explicaciones visuales" sobre las decisiones de los modelos basados en redes neuronales convolucionales. Dada una imagen como entrada, se pasa la imagen por la red, como parte de los cálculos *forward* y se produce un *output*. Luego, se toma una clase objetivo de salida del modelo, se calculan las derivadas de esta clase objetivo con respecto a las capas intermedias (convolucionales) hasta una capa seleccionada de la red, usando *back propagation*, hasta la capa objetivo. Se obtienen varios valores intermedios que se combinan haciendo *average pooling*, se los pasa por una función de activación ReLu (ya que solo nos interesan las activaciones con signo positivo) y se forma una nueva imagen con estos valores. Es uno de los métodos más utilizados para producir explicaciones visuales y, a diferencia de algoritmos como CAM (Zhou et al., 2016), no necesita reentrenar el modelo para ser utilizado. Partiendo de la base de que no es necesario reentrenar al modelo (buscamos medir la performance del modelo en una situación similar a la que puede ser utilizado en la práctica), y los trabajos relacionados que utilizaron el mismo método (Pillai y Pirsiavash, 2021; Schaaf et al., 2021), es que elegimos este algoritmo para

el presente trabajo.

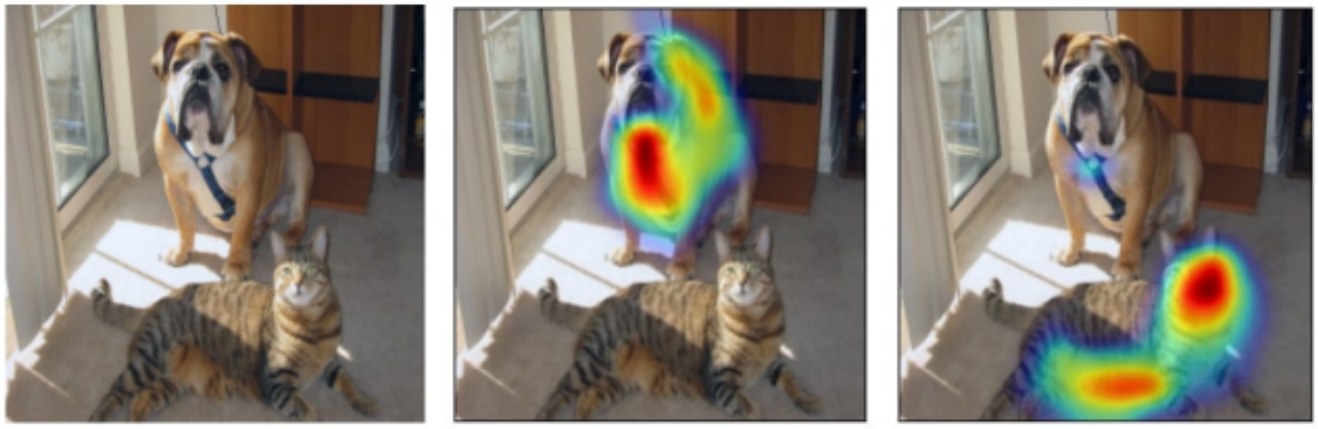


Figura 2.4: Ejemplo de activaciones de Grad-CAM para las clases *Dog* y *Cat* (Selvaraju et al., 2020, Figure 3)

2.6. Modelo

En este estudio, se buscó un modelo de clasificación de imágenes que sea ampliamente utilizado y que haya obtenido una *performance* considerada estado del arte para esta tarea. La idea fue el utilizar un modelo que haya sido preentrenado sobre el conjunto de datos ImageNet (Deng et al., 2010) y continuar el entrenamiento sobre la tarea específica de este trabajo (*Transfer Learning*). Se continuó el entrenamiento utilizando la función de pérdida entropía cruzada (Sección 2.7), ampliamente utilizada para la tarea de clasificación de imágenes, como así también la nueva función de pérdida aquí propuesta. Se buscó generar resultados comparables y el poder aislar el aporte específico de la función de pérdida propuesta (como se analizará en la Sección 2.8).

Dado los requisitos aquí expuestos, se utilizó el modelo ResNet50 (He et al., 2015). Este se encuentra disponible en la mayoría de las librerías más comunes de aprendizaje profundo. Se utilizó su versión preentrenada en las imágenes de ImageNet (Deng et al., 2010), mediante la librería PyTorch (Paszke et al., 2019).

Se cambió la última capa de la red, para que, en vez de que tenga una salida para las 1000 clases de ImageNet (Deng et al., 2010), tenga solo 4 clases (las 3 clases mencionadas en la Sección 3.1, más una, para cuando no detecta ningún objeto). Luego, se realizó un re entrenamiento completo de la red (sin congelar ningún peso) para las clases y la tarea específica a clasificar, aplicando una penalización a las activaciones de Grad-CAM que ocurrieron fuera de las cajas delimitadoras. Los detalles de la construcción de la red y el esquema de entrenamiento se tratarán con mayor amplitud en la sección Sección 3.3.

2.7. Función de pérdida entropía cruzada

Parte de la función de pérdida propuesta incluye la función entropía cruzada. Está función de pérdida puede ser utilizada para un modelo de clasificación que tenga como salida una probabilidad entre 0 y 1. El valor de entropía cruzada aumenta cuando la probabilidad predicha se aleja del valor real de la etiqueta. Un modelo perfecto va a tener una entropía cruzada de 0. Cuando esta función de pérdida se utiliza en un problema multiclase, la fórmula es la siguiente (Cybenko et al., 1999):

$$EC = -\sum_{c=1} y_{o,c} \log(p_{o,c})$$

donde:

EC	es entropía cruzada
log	es el logaritmo natural
y	es el indicador binario (0 o 1) de si la etiqueta corresponde a la clase c
p	probabilidad predicha de que la observación o es de la clase c

2.8. Funciones de pérdida penalizando activaciones espurias

Para entrenar los clasificadores de imágenes, se utilizó la siguiente función de pérdida:

$$Pérdida = lce(f, x, y) + \lambda |g_c^y - \tilde{g}_c^y| 1$$

donde:

lce	corresponde a la entropía cruzada explicada previamente
λ	es el factor de penalización
g_c^y	son las activaciones totales
$\tilde{g}_c^y$	son las activaciones dentro de la caja delimitadora del objeto

El primer término corresponde a la función de entropía cruzada tradicional (Sección 2.7) que incentiva al modelo a predecir la clase correcta. El segundo término incentiva a mantener la consistencia de la interpretabilidad del modelo, tomando como valor base todas las activaciones que pasaron dentro de la imagen para la clase correcta a predecir y restando las activaciones contenidas dentro del *bounding box*. Este término es multiplicado por λ, el cual es un hiperpárametro a seleccionar. Un valor más alto de λ significara una mayor importancia de la penalización propuesta, mientras que un valor menor sera lo contrario. En caso de que el modelo haya tenido todas sus activaciones dentro del *bounding box*, el resultado del segundo término será 0 (no se penalizan las activaciones ocurridas fuera). En el caso de que hayan activaciones de Grad-CAM que deban ser penalizadas, este segundo término, antes de ser multiplicado por λ, es el error promedio absoluto entre las activaciones que pasaron dentro de los *bounding box* y las que ocurrieron en la imágen completa. Estas activaciones computadas mediante Grad-CAM tienen un valor de entre 0 y 1, por lo que esta diferencia se encuentra en el mismo rango. Esto implica que la entropía cruzada y la penalización de las activaciones espurias se encuentran en el mismo rango de valores, lo que

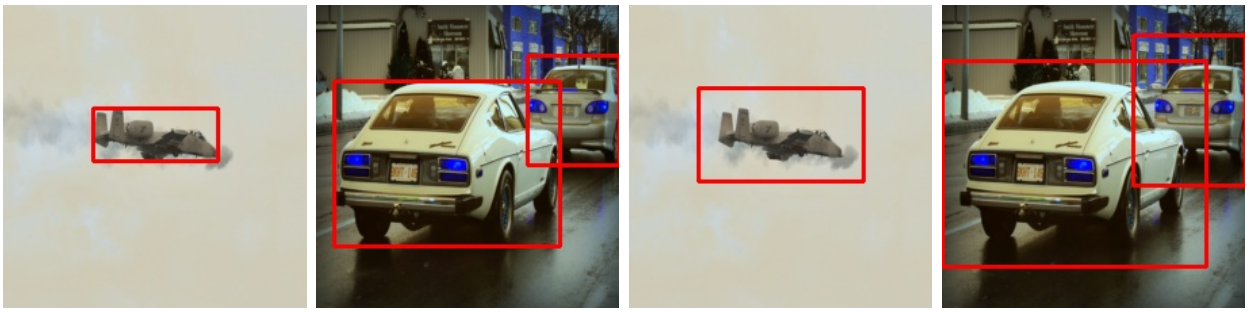
luego puede cambiar con la selección de hiperparamétros. Para el modelo base, contra el cual comparamos nuestros resultados (con el fin de analizar la utilidad de lo aquí propuesto), implementamos un valor de λ de 0, que equivale a un modelo entrenado solamente con la función de pérdida entropía cruzada.

Capítulo 3

Experimentos

3.1. Conjunto de datos

En el presente trabajo entrenamos un modelo para la tarea de clasificación de imágenes (indicar qué objetos aparecen dentro de la imágen). Dada la función de pérdida utilizada, donde se penalizan las activaciones no enfocadas en los objetos, se necesitaba contar con información sobre la localización de los objetos dentro de las imágenes. Existen diversos conjuntos de datos públicos que contienen imágenes y cajas delimitadoras sobre los objetos. Estos conjuntos de datos son normalmente utilizados para otra tarea conocida como *object localization*, que consiste en predecir la ubicación de los objetos, pero aun así funcionan perfectamente para nuestra tarea. Se buscó un conjunto de datos que incluya imágenes comúnmente utilizadas para entrenar clasificadores de imágenes y cuadros delimitadores. En un primer momento, se consideró el utilizar el conjunto de datos ImageNet (Deng et al., 2010), que cumple con los objetivos descritos y es comúnmente utilizado para medir avances en los métodos de aprendizaje automático. No obstante, se decidió utilizar una red ResNet50 en la librería PyTorch y utilizar la versión preentrenada en ImageNet (en función del *hardware* limitado disponible para realizar este trabajo). Esto generaba la posibilidad de tener solapamiento entre los datos utilizados para preentrenar estos modelos y en los experimentos llevados acabo aquí. Otros inconvenientes considerados fueron el tamaño del set de datos (1,28 millones de imágenes de entrenamiento para ImageNet2012), la cantidad



(a) Margen 0px (b) Margen 0px (c) Margen 15px (d) Margen 15px

Figura 3.1: Imágenes con sus *bounding box*, con y sin un margen extra sobre el mismo

de clases existentes (1000) y la dificultad para limitar la descarga de las imágenes correspondientes a una lista reducida de clases. Dadas estas limitaciones, se optó por utilizar OpenImagesV6 (Kuznetsova et al., 2020), con el cual evitamos el riesgo de solapamiento, y que brinda diversas facilidades para descargar solo las clases seleccionadas.

Se decidió descargar las imágenes correspondientes a las clases *Plane, Ship* y *Car*. Estas fueron elegidas buscando maximizar la diferencia del contexto entre cada clase y para reducir la posibilidad de que imágenes de un tipo de objeto aparezcan en conjunto con las imágenes de un objeto diferente. Para acceder a estos datos, se implementó, como es recomendado en el sitio web de OpenImages, la herramienta FiftyOne (Fif, 2022). OpenImages cuenta con sus ejemplos ya divididos en subconjuntos de entrenamiento, validación y testeo. Se utilizaron los subconjuntos originales y no fue necesario realizar ningún *split* extra sobre el conjunto de datos.

Muchas veces los *bounding box* no están perfectamente alineados con los objetos, lo que puede llevar a que realicemos una penalización considerando que una activación ocurrió fuera del objeto cuando no fue así. Para sobreponernos a este problema, se probó el agregar un margen extra a cada *bounding box* (agrandando el área cubierta por el mismo), como se ve en la Figura 3.1. Se compararon resultados utilizando este margen extra como así también utilizando los *bounding box* originales.

3.2. Métricas

Dado el objetivo de lograr modelos más robustos con el método de entrenamiento propuesto, necesitamos definir una métrica que nos sirva para cuantificar nuestros resultados. Es importante destacar que podríamos lograr un modelo más robusto, con menos activaciones fuera de los *bounding box*, sin que mejore el *accuracy* (precisión). Se obtendría un modelo más enfocado en los objetos a predecir manteniendo una precisión similar. En nuestro método de entrenamiento, la información que se está penalizando, sirve para predecir correctamente la clase. Lo esperable es que el modelo vea reducida su precisión, pero que mejore su robustez en las situaciones que el objeto aparece sin este contexto. Se eligió optimizar una métrica que sea descriptiva, de qué parte del área de activación corresponde realmente al objeto a predecir, por lo cual se implementó una métrica similar al Content-Heatmap propuesto por Pillai y Pirsiavash (2021). Se generó una nueva métrica que calcula el porcentaje de la activación de Grad-Cam contenido dentro del *bounding box*. Dado lo explicado en la Sección 3.1, se probó generar un margen alrededor de los *bounding box*. Esto permitió reducir la penalización en casos donde este no está perfectamente alineado con los bordes del objeto. Los resultados de los experimentos con y sin margen están explicados en la sección de resultados, en donde medimos los valores de Content-Heatmap y de *accuracy* para cada modelo resultante.

$$ContentHeatmap = \frac{\sum_{n=1}^{BS} areaGC \triangle BBOX}{\sum_{n=1}^{BS} areaGC \triangle IMG}$$

donde:

BS	es el tamaño de lote
GC	es Grad-CAM
BBOX	es cuadros delimitadores
IMG	es la imágen completa

3.3. Detalles de la implementación

En el análisis aquí desarrollado, se utilizó el modelo ResNet50 preentrenado en ImageNet, por lo cual fue necesario cambiar la salida de la red, para que en vez de que devuelva las 1000 clases de ImageNet, devuelva las 4 clases en las que nos enfocamos.

En conjunto con este modelo, se utilizó Grad-CAM, cuya implementación se hizo con la librería pytorch-grad-cam, con el fin de poder detectar las regiones que causaron las activaciones de la red (y poder penalizarlas cuando corresponda). Dado que Selvaraju et al. (2020) recomienda utilizar la última capa convolucional del modelo, en este caso, usando ResNet50, la capa utilizada fue la *layer 4*.

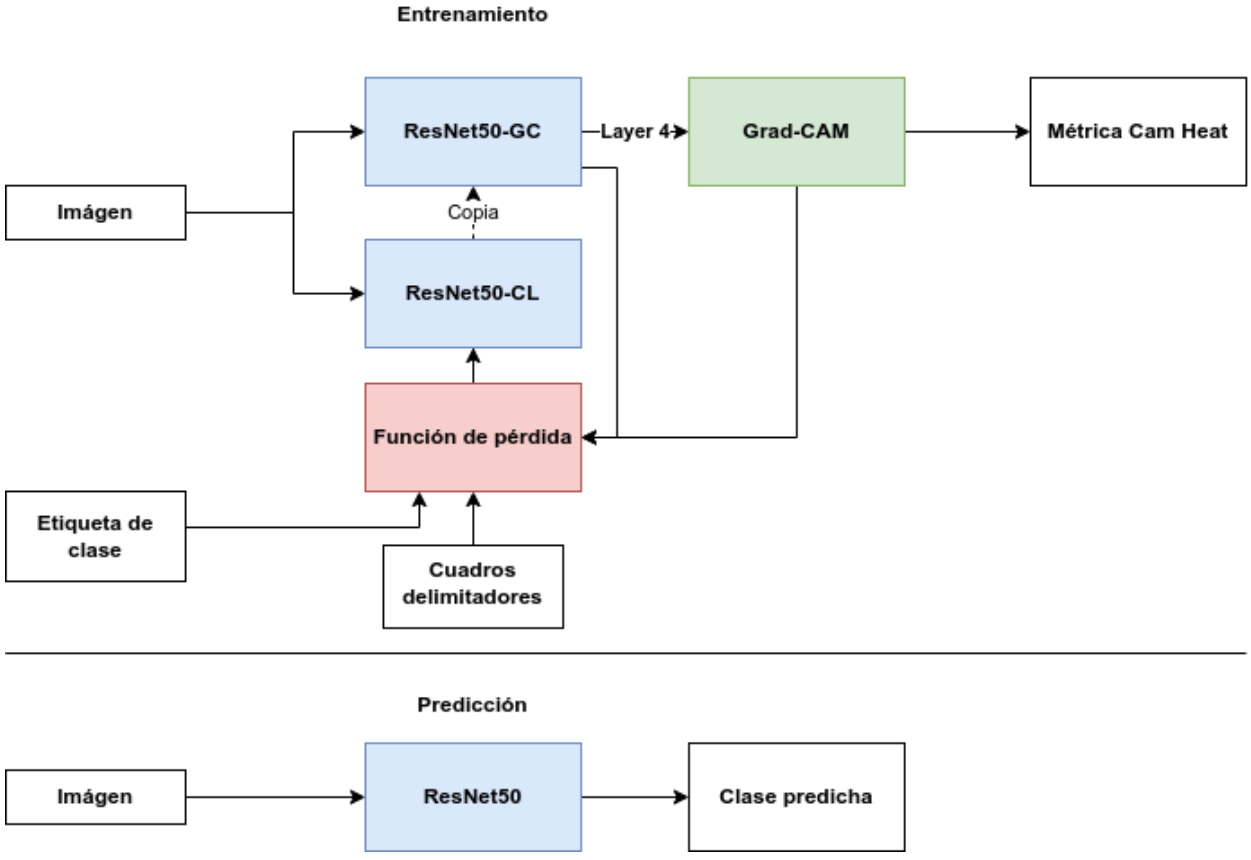


Figura 3.2: Implementación

Nuestra implementación completa puede ser visualizada en la Figura 3.2. En esta, vemos todos los componentes que forman nuestro proceso donde una imágen es la entrada a la

red ResNet50. Sobre este se realiza el proceso necesario para obtener la visualización de Grad-CAM. La salida original de ResNet50, sumada a la visualización de Grad-CAM y la información de los cuadros delimitadores, es ingresada a la función de pérdida. Luego, con la función de pérdida, se realiza *back-propagation* sobre un modelo ResNet50 copiado del objeto original (a continuación se explica en más detalle esta decisión). El motivo por el cual existen dos objetos ResNet50, que de aquí en adelante llamaremos ResNet50-CL (clasificador) y ResNet50-GC (Grad-CAM), es que uno se utiliza para obtener los gradientes de Grad-CAM y el otro se utiliza para obtener los gradientes necesarios para aprender mediante *back-propagation*.

El proceso de Grad-CAM implica realizar *back-propagation* para visualizar las activaciones. Los gradientes calculados para el proceso de Grad-CAM se sobreponen a los gradientes a calcular con nuestra función de pérdida. Esto significa que el modelo vea afectado sus pesos para la obtención de los valores de Grad-CAM. El modificar los pesos del modelo sin contrastar esta información con ninguna etiqueta externa no es un comportamiento deseado. Con el fin de sobreponernos a esta situación, se realiza una copia del objeto del modelo en cada paso del entrenamiento. De esta forma, realizamos Grad-CAM sobre el modelo ResNet50-GC, pero aplicamos nuestra función de pérdida al modelo ResNet50-CL (no afectado por el cálculo de Grad-CAM). Al final de cada *step* de entrenamiento, se vuelve a realizar la copia, para que la red afectada por Grad-CAM ResNet50-GC sea reemplazada por la versión más reciente de ResNet50-CL. Al momento de la predicción, sólo es utilizado el modelo ResNet50.

En linea con lo recién explicado, un *step* de entrenamiento en nuestra arquitectura consiste en 1)obtener los gradientes de Grad-CAM sobre el modelo ResNet50-GC 2)obtener las predicciones de clase del modelo ResNet50-CL 3)calcular la pérdida entropía cruzada 4)calcular la pérdida que penaliza las activaciones espurias (tomando como entrada los gradientes Grad-CAM y el *bounding box* donde se encuentra el objeto) 5)sumar ambas perdidas con sus respectivos factores (como es explicado en Sección 2.8) 6)realizar *back-propagation*

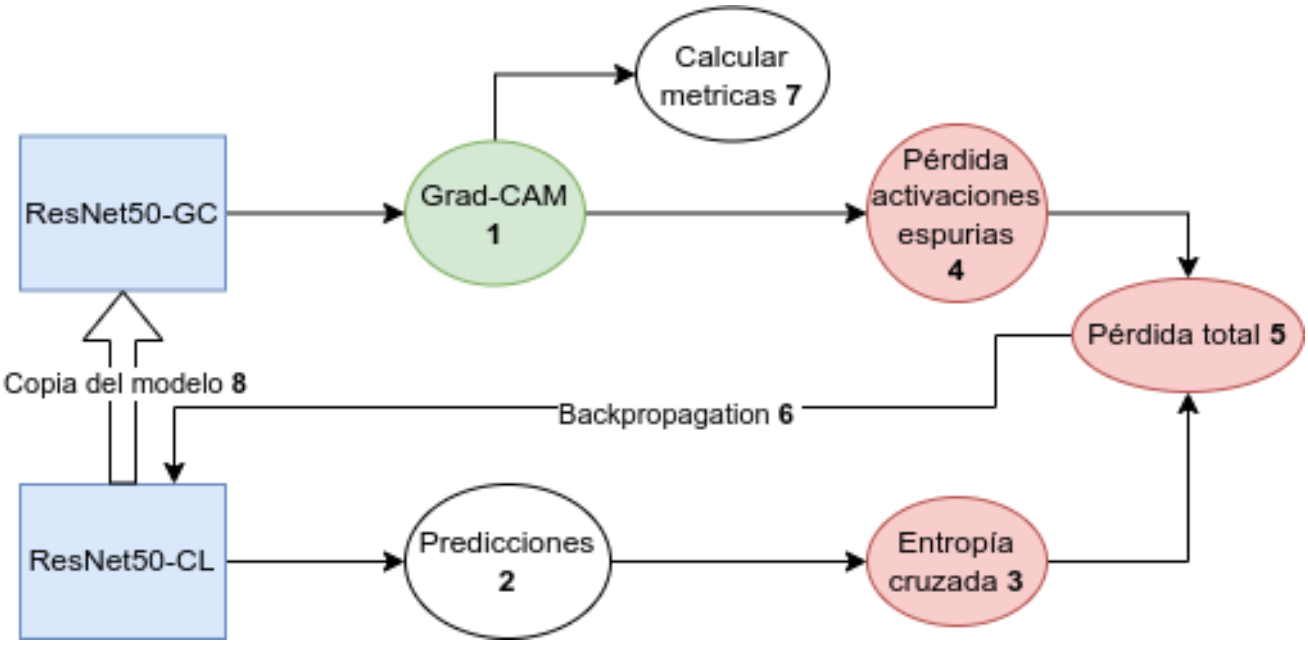


Figura 3.3: Entrenamiento

sobre ResNet50-CL 7)computar las métricas correspondientes 8)duplicar el objeto del modelo (copiamos ResNet50-CL a ResNet50-GC). Este proceso puede ser visualizado en la Figura 3.3.

El *hardware* utilizado para este proyecto fue un procesador Intel I9, 48gb de *RAM* y una Nvidia Quadro RTX 4000 con 8gb de memoria. Dado la limitación de los recursos disponibles, mayormente en la cantidad de memoria de la GPU, se limitó la cantidad de imágenes a 10.000 para entrenamiento, a 2.500 para validación y a 1.000 para testeo. En todos los casos se fijó una semilla dentro de la librería FiftyOne (Fif, 2022) para que los datos utilizados sean siempre los mismos. Todos los experimentos fueron corridos por 30 épocas, con un tamaño de lote de 16. Se utilizó el modelo, que dentro de estas 30 épocas, tuvo la mejor *performance* dentro de la métrica Content Heatmap sobre los datos de validación.

3.4. Resultados

λ	Acc train	Acc val	Cam Heat val
0 (Solo CE)	97.33	97.72	69.20
1	**99.22**	96.68	71.15
10	98.52	96.88	70.04
25	98.24	**97.76**	**71.68**

Cuadro 3.1: Modelos entrenados y probados con margen 0 sobre los cuadros delimitadores

λ	Acc train	Acc val	Cam Heat val
0 (Solo CE)	**99.29**	97.48	82.89
1	98.96	**97.76**	83.60
10	99.02	97.44	**83.88**
25	99.16	97.72	83.29

Cuadro 3.2: Modelos entrenados y probados con margen 15 sobre los cuadros delimitadores

En el cuadro 3.1 vemos los resultados obtenidos sin el margen sobre los cuadros delimitadores. En el cuadro 3.2 vemos los resultados con un margen de 15px. El parámetro lambda controla cuánto se penalizan sobre las activaciones espurias. En el caso particular en el que no se consideran estas activaciones el parámetro lambda sería 0. Es decir, funciona como entropía cruzada tradicional. En ambos casos vemos una mejora sobre la métrica Content Heatmap al utilizar la función de pérdida propuesta por este trabajo con todos los valores probados de λ mayores a 0.

En el primer caso, sin margen en los cuadros delimitadores, dentro los distintos parámetros probados, y basado en la métrica Content Heatmap sobre los datos de validación, vemos los mejores resultados utilizando λ de valor 25, con el valor de 71.68, comparado con 69.20 (2.38 puntos extra) de nuestro modelo base. También así, obtenemos el mejor valor de *accuracy* sobre los datos de validación con λ 25 (con una diferencia mínima comparado con los resultados con λ 0). Esta función de pérdida penaliza las activaciones que pasan fuera de la caja delimitadora, pero no sería esperable que mejore significativamente el *accuracy*

comparado con un modelo entrenado solo con entropía cruzada.

Cuando aplicamos un margen sobre las cajas delimitadoras, como vemos en los resultados descritos en el cuadro 3.2, vemos que el mejor resultado de Content Heatmap, para datos de validación, está dado cuando se utiliza un λ de 10. En este caso, obtuvimos 83.88, comparado con 82.89 del modelo solo entrenado con entropía cruzada (0.99 puntos extra). Esto parecería sugerir que al minimizar la posible pérdida calculada (ya que permitimos que una mayor área de las activaciones de Grad-CAM sea incluida dentro de una caja delimitadora más grande) necesitamos realizar una penalización menor para lograr que el modelo se enfoque en las áreas pertenecientes al objeto a predecir.

λ	Marg. entr.	Acc test	Cam Heat test
0 (Solo CE)	0px	97.40	67.84
1	0px	96.40	69.91
10	0px	96.60	68.85
25	0px	**98.40**	70.53
1	15px	98.10	69.78
10	15px	97.40	**71.31**
25	15px	98.10	69.81

Cuadro 3.3: Modelos probados con margen 0 sobre los cuadros delimitadores, datos de testeo

Estos resultados, donde el mejor modelo entrenado sin margen sobre los *bounding box* fue el que utilizó λ 25, y cuando entrenado con margen de 15px sobre los *bounding box*, el modelo con λ 10, no son directamente comparables, ya que los valores de Content Heatmap son más altos siempre cuando se prueben los datos con un márgen extra sobre las cajas delimitadoras (lo que es esperable, ya que nuestra prueba es más permisiva). Con el fín de poder determinar cual es el modelo que performa mejor y validar el aporte de entrenar el modelo con un márgen extra sobre las cajas delimitadoras, se probaron todos los modelos ante los datos de testeo sin usar un margen extra al momento de computar las métricas. Esto puede ser visualizado en la tabla 3.3. En esta situación el modelo que otorga mejores resultados es el que utilizó un λ 10 y fue entrenado con un margen de 15px. En ese caso, obtenemos un Content Heatmap de 71.31, superior al valor de 67.84 que obtuvo el modelo entrenado con λ 0 (solo CE), y

superior también al de 70.53 que obtuvo el modelo con mejores resultados habiendo sido entrenado sin utilizar un margen extra sobre las cajas delimitadores (utilizando un λ de 25), lo que es una mejora de 3.47 puntos sobre el modelo base y de 0.78 puntos sobre el modelo no entrenado con márgen extra. Esto nos sugiere que el modelo se beneficia de este margen extra sobre los cuadros delimitadores, posiblemente porque nos ayuda a penalizar las activaciones que realmente son dadas por el contexto, y no aquellas que se encuentran muy próximas al objeto a predecir (lo cual podría, incluso, estar dado por un *bounding box* no alineado perfectamente). Es preciso también destacar, que la métrica *accuracy* tuvo una mejora no despreciable medida con los datos de testeo en los modelos que utilizaron λ 25 y fueron entrenados sin margen extra sobre los cuadros limitadores (98.40 *accuracy* medido contra 97.40 del baseline), como así también en los modelos con λ 1 y 25 cuando son entrenados con un margen extra sobre los cuadros delimitadores (98.10 en ambos casos, medido contra 97.40 del baseline). Esto nos plantea que el método de entrenamiento propuesto por este trabajo podría tener un beneficio no solo con el fin de reducir el sesgo, sino también lograr un modelo que performe mejor en cuanto a los resultados de sus predicciones sin importar la causa.

A modo de ejemplos cualitativos se incluyen algunas imagenes de las activaciones y las métricas en el cuadro 3.4. Allí vemos que el modelo donde se aplicó λ 25 durante su entrenamiento (basado en lo que vemos con Grad-CAM), se enfoca en los múltiples automóviles que aparecen en la imagen, mientras que el modelo con λ 0 solo se enfoca en el automóvil que está más grande y cerca en la imagen. Esto, en el ejemplo observado, se traduce en una diferencia de 8.49 puntos en la métrica de Content Heatmap.

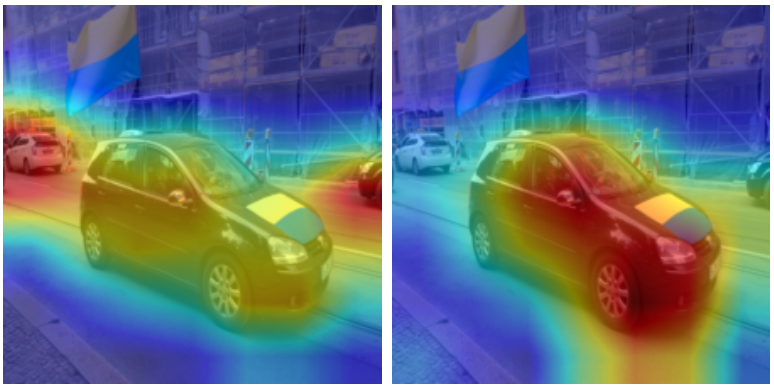
λ	Imagen	Cam Heat calculado
0 (Solo CE)		65.28
25		73.77

Cuadro 3.4: Imágenes pertenecientes a conjunto de datos de testeo

3.5. Limitaciones

Entendemos que hay distintas limitaciones en este trabajo, entre las cuales se puede mencionar la cantidad de clases elegidas para los tipos de objetos a detectar, la cantidad de imágenes utilizadas para el entrenamiento (dado el *hardware* disponible), las métricas utilizadas y como fue creado el conjunto de datos utilizado. Vamos a enfocarnos en estas dos últimas ya que presentan una dificultad mayor para su resolución.

Como se vio previamente, la métrica principal utilizada fue Content Heatmap (Sección 3.2). Como se explica en la sección respectiva, computa el área de activaciones que suceden dentro del área en la cual se indica que se encuentra el objeto (mediante cajas delimitadoras). Esta forma de computar los resultados implica que si el modelo produce una pequeña activación, pero la misma se encuentra dentro de la caja delimitadora del objeto, obtendremos un valor de Content Heatmap de 100. En el caso de un modelo que produce muchas activaciones dentro de la caja delimitadora, pero una pequeña parte se encuentra fuera de los límites de la caja, su valor de Content Heatmap será necesariamente menor, aún



(a) λ 10, CH = 64.00 (b) λ 0 (CE), CH = 65.25

Figura 3.4: Resultados sobre imágenes de testeo, donde un Content Heatmap superior no implica un mejor rendimiento del modelo

si este modelo funciona mejor en la práctica (en cuanto a que las activaciones se enfoquen en el área de la imagen que contiene al objeto). En el cuadro 3.4 vemos un ejemplo de esto, donde el modelo que resulta con un valor de Content Heatmap superior no produce activaciones alrededor del automóvil que se ve en la esquina superior izquierda, mientras que el modelo que si tiene estas activaciones, y parece hacer un mejor trabajo (en cuanto a activar las regiones relacionadas a los objetos a detectar), obtiene un valor de Content Heatmap inferior. Esto es una limitación de la métrica y, en un trabajo futuro, debería ser revisitado.

Por otro lado, como es explicado por Gan (2022), el *set* de datos implementado tiene sus limitaciones, en cuanto a que existen cajas delimitadoras que están puestas en el lugar incorrecto, tienen el tamaño incorrecto, o no están presente cuando deberían estarlo. Así también, hay muchas clases que pueden ser utilizadas para el mismo objeto y por la naturaleza de cómo se creó el conjunto de datos (Kuznetsova et al., 2020), este criterio puede cambiar de imagen a imagen. Por ejemplo, en algunos casos, podemos tener una imágen de una camioneta Pick-up que tenga asignada la etiqueta camioneta (*Truck*), pero en otros, este mismo vehículo, puede ser etiquetado como automóvil (vemos un ejemplo de esto en el cuadro 3.5). En algunos casos, esto puede traducirse en que el entrenamiento propuesto se realice de forma incorrecta. Por ejemplo, podría ser que penalicemos a un modelo por

generar una activación de un automóvil fuera de la caja delimitadora indicada para el objeto, pero, en realidad, el modelo está activándose en el lugar correcto, solo que la caja delimitadora está asignada a una clase distinta. Este mismo ejemplo se podría aplicar al momento de computar las métricas de validación o de testeo, generando que nuestro resultado sea menor que el que realmente es (el modelo está en lo correcto, pero por la falta de una caja delimitadora, computamos un valor de Content Heatmap más bajo que el real). Es esperable que estos errores sean relativamente pequeños y no estén presentes en la mayoría de los casos.

Figura 3.5: Imágenes pertenecientes a OpenImages-V6 en su conjunto de Validación-Testeo. En la imagen de la izquierda vemos que el vehículo marron no cuenta con la etiqueta Car, por ser una camioneta, mientras que en la imagen de la derecha, otra camioneta, contiene la etiqueta Car

3.6. Conclusiones

Nuestro objetivo era el identificar un método de entrenamiento que nos permita guiar las activaciones de un modelo hacía las áreas donde se encuentran los objetos. Hemos alcanzado este objetivo obteniendo mejores resultados que la base con la que fue comparado cuando se midió por la métrica Content-Heatmap (utilizando la salida del algoritmo Grad-CAM). Esto significa que nuestro modelo está más enfocado en las regiones donde realmente se encuentran los objetos a predecir. El contar con explicaciones visuales más consistentes es de suma importancia, especialmente cuando los modelos son utilizados en áreas críticas y esta explicación visual es utilizada por el usuario para realizar un diagnóstico, como puede ser en imágenes médicas. Así también, en algunos casos, se logró una mejora de la métrica *accuracy* (cuadro 3.3), lo cual validaria que el entrenamiento propuesto le aporta información útil al modelo, aun en los casos donde solo se esta midiendo los resultados para la tarea de clasificación, y el sesgo no está siendo relevado. Nuestro modelo debería ser más robusto en poder detectar los objetos para los que fue entrenado, aún en condiciones distintas que los datos de su entrenamiento (y vemos algunos ejemplos como en el cuadro 3.4). Con la ampliación de las áreas de uso de los algoritmos de inteligencia artificial, es de suma importancia que estos puedan adaptarse a la mayor cantidad de escenarios posibles, aún cuando son utilizados en ambientes diferentes a los de su entrenamiento. Por ejemplo, un modelo utilizado en un vehículo autónomo, debe poder reconocer a un ciclista, aún si el mismo se encuentra en una situación atípica o fuera de contexto.

3.7. Trabajos futuros

Si bien se obtuvieron excelentes resultados bajo las métricas explicadas en este trabajo, la relación entre las reales causas de la activación y lo que nos muestra Grad-CAM (y otros algoritmos de explicabilidad), es un motivo de discusión (Schaaf et al., 2021). Por esto, es necesario la realización de una cantidad mayor de experimentos para validar la consistencia de estos resultados. Una exploración posible sería entrenar el modelo con la función de pérdida aquí propuesta y Grad-CAM, y validar estos resultados utilizando un algoritmo de explicabilidad diferente (para determinar si los mejores resultados de la métrica Cam-Heat se repiten en otros algoritmos).

En caso de contar con mayores recursos de procesamiento, sería útil incluir el conjunto de datos completo (en contraposición el conjunto reducido utilizado en este trabajo) y probar de entrenar el modelo desde 0 (en vez de utilizar el modelo preentrenado en ImageNet). Así también, sería favorable explorar otras métricas que puedan ser indicativas del área de activación de la red y que puedan sobreponerse a las limitaciones de las métricas utilizadas en este análisis. Por otro lado, sería útil el generar un conjunto de datos sintético, donde se presenten situaciones muy atípicas, como un automóvil sobre el agua, buscando generar situaciones que no estén representadas en el conjunto de datos utilizado durante el entrenamiento y medir la *performance* del modelo aquí planteado sobre esas imágenes.

En el conjunto de datos, se decidió utilizar clases muy distintas entre sí (*Plane, Ship* y *Car*), ya que puede ser problematico si dos objetos de clases distintas aparecen en la misma imágen y el modelo tiene que predecir sólo una. Por otro lado, por la naturaleza de cómo fue creado OpenImages, podemos tener dos clases válidas para el mismo objeto. Si se cuenta con mayores recursos y se logra estandarizar las etiquetas de clase, sería interesante medír la *performance* del método propuesto en un modelo que utilice clases muy cercanas (como puede ser Auto, Moto y Peaton) y ver si lo aquí propuesto logra evitar errores comunes en la tarea de clasificación de imágenes.

Los modelos que utilizan capas del tipo *Attention* (Vaswani et al., 2017) están siendo

ampliamente utilizados para el procesamiento de lenguaje natural, así también se demostró la utilidad de este tipo de capa para las tareas de visión por computadora (Dosovitskiy et al., 2020). En la implementación de Dosovitskiy et al. (2020), se divide la imagen en una secuencia de recortes, que luego es ingresada como entrada a la red con el fín de que las capas de *Attention* aprendan a darles distintos pesos a los distintos recortes. El tener estos recortes y las capas *Attention* permitiría penalizar más directamente las activaciones en los recortes donde no se encuentra el objeto a predecir (que se puede determinar utilizando las cajas delimitadoras). Sería muy interesante adaptar la propuesta del presente trabajo a ese tipo de arquitectura, y comprobar si esta propuesta puede ser aún más efectiva en ese caso.

Bibliografía

(2011). ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

(2022). FiftyOne — FiftyOne 0.18.0 documentation.

(2022). I performed Error Analysis on Google's Open Images dataset and now I have trust issues | by Tyler Ganter | Towards Data Science.

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. E., y Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey.

Bany Muhammad, M. y Yeasin, M. (2021). Eigen-CAM: Visual Explanations for Deep Convolutional Neural Networks. *SN Computer Science 2021 2:1*, 2(1):1–14.

Bishop, C. M. (1998). Neural networks and their applications. *Review of Scientific Instruments*, 65(6):1803.

Burkart, N. y Huber, M. F. (2021). A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research*, 70:245–317.

Choi, J., Gao, C., Messou, J. C., y Huang, J. B. (2019). Why Can't I dance in the mall? learning to mitigate scene bias in action recognition. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation.

Clark, C., Yatskar, M., y Zettlemoyer, L. (2020). Learning to Model and Ignore Dataset Bias with Mixed Capacity Ensembles. *EMNLP 2020*, pages 3031–3045.

Cybenko, G., O'Leary, D. P., y Rissanen, J. (1999). *The mathematics of information coding, extraction, and distribution*. Springer.

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, y Li Fei-Fei (2010). ImageNet: A large-scale hierarchical image database. pages 248–255. Institute of Electrical and Electronics Engineers (IEEE).

Devlin, J., Chang, M. W., Lee, K., y Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., y Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Feng, V. (2017). An Overview of ResNet and its Variants | by Vincent Feng | Towards Data Science.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics 1980 36:4*, 36(4):193–202.

He, K., Zhang, X., Ren, S., y Sun, J. (2015). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778.

Hubel, D. H. y Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.

Ibrahim, R. y Shafiq, M. O. (2022). Augmented Score-CAM: High resolution visual interpretations for deep neural networks. *Knowledge-Based Systems*, 252:109287.

Iosifidis, V. y Ntoutsi, E. (2018). Dealing with bias via data augmentation in supervised learning scenarios. *CEUR Workshop Proceedings*, 2103:24–29.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., y Sayres, R. (2017). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *35th International Conference on Machine Learning, ICML 2018*, 6:4186–4195.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., y Ferrari, V. (2020). The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *International Journal of Computer Vision*, 128(7):1956–1981.

LeCun, Y., Bottou, L., Bengio, Y., y Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323.

Li, Y. y Vasconcelos, N. (2019). Repair: Removing representation bias by dataset resampling.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., y Galstyan, A. (2019). A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35.

Minsky, M. y Papert, S. (1969). Perceptrons: an introduction to computational geometry. page 258.

Nair, V. y Hinton, G. E. (2010). Rectified linear units improve Restricted Boltzmann machines. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 807–814.

Nuriel, O., Benaim, S., y Wolf, L. (2020). Permuted AdaIN: Reducing the Bias Towards Global Statistics in Image Classification. *CVPR 2021*, pages 9482–9491.

Oh, K. S. y Jung, K. (2004). GPU implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., y Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation.

Pillai, V., Koohpayegani, S. A., Ouligian, A., Fong, D., y Pirsiavash, H. (2021). Consistent Explanations by Contrastive Learning.

Pillai, V. y Pirsiavash, H. (2021). Explainable Models with Consistent Interpretations. In *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, volume 3B, pages 2431–2439.

Ravishankar, H., Sudhakar, P., Venkataramani, R., Thiruvenkadam, S., Annangi, P., Babu, N., y Vaidya, V. (2016). Understanding the mechanisms of deep transfer learning for medical images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10008 LNCS, pages 188–196. Springer Verlag.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Ruder, S., Peters, M. E., Swayamdipta, S., y Wolf, T. (2019). Transfer Learning in Natural Language Processing. *Proceedings of the 2019 Conference of the North*, pages 15–18.

Schaaf, N., de Mitri, O., Kim, H. B., Windberger, A., y Huber, M. F. (2021). Towards Measuring Bias in Image Classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12893 LNCS, pages 433–445.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., y Batra, D. (2020). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2):336–359.

Singh, K. K., Mahajan, D., Grauman, K., Lee, Y. J., Feiszli, M., y Ghadiyaram, D. (2020). Don't Judge an Object by Its Context: Learning to Overcome Contextual Bias. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 11067–11075.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., y Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 1–9. IEEE Computer Society.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., y Fergus, R. (2013). Intriguing properties of neural networks.

Szeliski, R. (2022). *Computer vision : algorithms and applications*.

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., y Schmidt, L. (2020). Measuring Robustness to Natural Distribution Shifts in Image Classification. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.*

Tommasi, T., Patricia, N., Caputo, B., y Tuytelaars, T. (2017). A deeper look at dataset bias. In *Advances in Computer Vision and Pattern Recognition*, number 9783319583464, pages 37–55. Springer London.

Tong, S. y Kagal, L. (2020). Investigating Bias in Image Classification using Model Explanations.

Torralba, A. y Efros, A. A. (2011). Unbiased look at dataset bias. In *Proceedings of the*

IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1521–1528. IEEE Computer Society.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017-December:5999–6009.

Voulodimos, A., Doulamis, N., Doulamis, A., y Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review.

Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., y Hu, X. (2019). Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2020-June, pages 111–119. IEEE Computer Society.

Weiss, K., Khoshgoftaar, T. M., y Wang, D. D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1):1–40.

Yamaguchi, K., Sakamoto, K., Akabane, T., y Fujimoto, Y. (1990). A neural network for speaker-independent isolated word recognition. *1st International Conference on Spoken Language Processing, ICSLP 1990*, pages 1077–1080.

Zhang, Q., Wang, W., y Zhu, S. C. (2018). Examining CNN representations with respect to dataset bias. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, volume 32, pages 4464–4473.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., y Torralba, A. (2016). Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 2921–2929.