GOBIERNO DE MÉXICO | CONAHCYT CONSEJO NACIONAL DE HUMANIDADES CIENCIAS Y TECNOLOGÍAS | INFOTEC

BIBLIOTECA INFOTEC
VISTO BUENO DE TRABAJO TERMINAL

Maestría en Ciencia de Datos e Información
(MCDI)

Ciudad de México, a 14 de junio de 2024

UNIDAD DE POSGRADOS
PRESENTE

Por medio de la presente se hace constar que el trabajo de titulación:

"Evaluación del desempeño de diferentes técnicas de aprendizaje computacional aplicadas a la clasificación de imágenes remotas"

Desarrollado por el alumno: **Helio Salvador Jiménez García**, bajo la asesoría de la **Dra. Tania Aglaé Ramírez del Real** cumple con el formato de Biblioteca, así mismo, se ha verificado la correcta citación para la prevención del plagio; por lo cual, se expide la presente autorización para entrega en digital del proyecto terminal al que se ha hecho mención. Se hace constar que el alumno no adeuda materiales de la biblioteca de INFOTEC.

No omito mencionar, que se deberá anexar la presente autorización al inicio de la versión digital del trabajo referido, con el fin de amparar la misma.

Sin más por el momento, aprovecho la ocasión para enviar un cordial saludo.

Mtro. Carlos Josué Lavandeira Portillo
Director Adjunto de Innovación y Conocimiento

CJLP/jah

C.c.p. Felipe Alfonso Delgado Castillo.- Gerente de Capital Humano.- Para su conocimiento.
Helio Salvador Jiménez García.- Alumno de la Maestría en Ciencia de Datos e Información.- Para su conocimiento.

MCDI
MAESTRÍA EN CIENCIA DE DATOS E INFORMACIÓN

INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

DIRECCIÓN ADJUNTA DE INNOVACIÓN Y CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS

“Evaluación del desempeño de diferentes técnicas de aprendizaje computacional aplicadas a la clasificación de imágenes remotas”

Implementación de Proyecto
Que para obtener el grado de MAESTRO EN CIENCIA DE DATOS E INFORMACIÓN

Presenta:

Helio Salvador Jiménez García

Asesor:

Dra. Tania Aglaé Ramírez del Real

Ciudad de México, Diciembre, 2023.

CONACYT

INFOTEC POSGRADOS

Tabla de contenido

Índice de figuras

Índice de tablas

Siglas y abreviaturas

ANN	Redes neuronales artificiales.
API	Interfaz de programación de la aplicación.
AVHRR	Descripción
BN	Normalización por lotes
BP	Propagación hacia atrás o retropropagación.
CH	Histograma de color.
CNN	Redes neuronales convolucionales.
CPU	Unidad central de procesamiento.
DBN	Redes profundas de creencia.
DS	Almacén de datos.
FC	Capa completamente conectada.
FLOP	Medida de la velocidad de una computadora, en operaciones aritméticas por segundo.
GB	Gigabyte.
GHz	Gigahertz.
GIST	Transformación global invariante a la escala de una imagen.
GPU	Unidad de procesamiento gráfico.
HOG	Histograma de gradientes orientados.
IA	Inteligencia artificial.
IKONOS	Satélite comercial construido para la observación terrestre con resolución entre 1 y 4 metros.
ILSVRC	Competencia de reconocimiento visual a gran escala basada en el conjunto de datos ImageNet.
Inception	Red neuronal convolucional desarrollada por Google, basada en la idea de redes dentro de redes.

JPEG	Método de compresión de imágenes digitales con pérdida de información.
LANDSAT	Conjunto de satélites para la observación y toma de imágenes remotas de la superficie terrestre.
LBP	Patrones binarios locales.
MB	Megabyte.
MLP	Perceptrón multicapa.
PCA	Análisis de componentes principales.
QUICKBIRD	Satélite comercial de alta resolución para el análisis de la superficie terrestre mediante imágenes.
RAM	Memoria de acceso aleatorio.
RBM	Máquinas Boltzmann restringidas.
ReLU	Unidad lineal rectificada.
ResNet	Redes residuales.
RGB	Siglas de los colores rojo, verde y azul.
RSICD	Base de datos que contiene imágenes remotas con textos descriptivos.
SAE	Auto codificadores apilados.
SGD	Descenso de gradiente estocástico.
SIFT	Transformación de características invariantes a la escala.
SPOT-5	Satélite Francés de alta resolución para la observación de la superficie terrestre
TIFF	Formato de archivo de imagen etiquetada. Se usa en el almacenamiento de imágenes basadas en píxeles.
TPU	Unidad de procesamiento de tensores.
UCM	Universidad de California sede Merced.
VGGNet	Red neuronal convolucional de grupo de geometría visual.

VHR Muy alta resolución.

Introducción

Desde la invención de la fotografía, a mediados del siglo XIX, se empezó a experimentar con imágenes tomadas desde gran altura, usando globos aerostáticos, o inclusive palomas. A principios del siglo XX, ya con la invención de los primeros aviones, se empezaron a tomar imágenes desde pleno vuelo sobre ciudades o regiones específicas.

Las fotografías aéreas fueron una herramienta muy valiosa durante la primera y segunda guerra mundial, donde se empleaban para el reconocimiento de objetivos militares [1]. Ya en la década de 1960, con la invención de los sistemas de radar y sensores multiespectrales; se obtuvieron imágenes en blanco y negro de fenómenos climatológicos. En la década de los 70, se iniciaron los lanzamientos de satélites con sensores multiespectrales como los LANDSAT y AVHRR, que se emplearon para monitorear condiciones climáticas, uso de suelo, entre otros [2].

Imágenes remotas y su aplicación.

El avance de la tecnología, ha permitido que las imágenes tomadas desde una gran altura hayan logrado un aumento en la resolución y en la cantidad de canales con información (imágenes remotas). Así tenemos como consecuencia que la utilidad de este tipo de fotografías es cada vez mayor en distintos campos de aplicación, como en el monitoreo ambiental, por ejemplo, cuando se trabaja en la extinción de un incendio de grandes proporciones que puede ser mejor monitoreado desde las alturas; o en la identificación del uso de suelo, al monitorear cómo evoluciona el crecimiento de las ciudades; en la agricultura, al identificar los diferentes tipos de siembras que se llevan a cabo en una región determinada, además de otras aplicaciones.

Con el concepto de imágenes remotas (RSI, por sus siglas en inglés), nos referimos al monitoreo de las condiciones de la superficie terrestre capturadas a través de fotografías tomadas desde gran altura por sensores multiespectrales colocados en satélites, aviones, drones o en vehículos no tripulados [3].

Otra utilidad de la clasificación de imágenes remotas es la vigilancia sobre la contaminación ambiental. Uno de los indicadores más relevantes para el monitoreo de la contaminación ambiental es la cantidad de partículas sólidas suspendidas en el aire que respiramos, conocida como PM_{10}. Son partículas suspendidas de 10 μm o menores en diámetro y que son dañinas para la salud, ya que tienen la posibilidad de introducirse en los pulmones al respirar el aire que contenga este tipo de partículas.

Existen imágenes remotas con canales espectrales adicionales y que capturan diferentes longitudes de onda del espectro radioeléctrico. En estos canales adicionales se captan las longitudes de onda capaces de detectar las partículas PM_{10} en el aire. Realizando la metodología de clasificación de imágenes remotas podemos detectar qué áreas contienen partículas PM_{10} y cuáles no; de tal forma que se podría identificar zonas de una ciudad con este tipo de contaminante [4].

Este gran avance en la tecnología de captación de imágenes ha mejorado la calidad y desempeño de las cámaras fotográficas colocadas en aviones, drones, vehículos no tripulados y mediante este medio capturar imágenes del terreno desde una gran altura. El área de superficie cubierta por este tipo de aviones es muy amplia y permite tomar grandes secciones de terreno a una gran definición, lo que nos proporciona información sobre todos los elementos sobre la superficie aunque sean pequeños. Esta gran cantidad de información generada por las imágenes remotas conlleva una gran cantidad de trabajo de procesamiento para

su análisis [5]. Cada vez su procesamiento es más complejo, tardado y costoso, ya que se ejecuta mediante la fuerza de trabajo de seres humanos.

Con los avances en las áreas de visión por computadora y aprendizaje computacional, se permite el procesamiento y análisis automatizado de una mayor cantidad de imágenes remotas que además presentan una menor incidencia de errores, mediante la implementación de una metodología que ayuda en la clasificación de imágenes remotas. Esta metodología se basa en el aprendizaje computacional para obtener las características más sobresalientes que las diferencian de las demás clases de imágenes.

Las imágenes se analizan mediante un conjunto de píxeles (unidad más pequeña de información en una imagen) que se distribuyen en un arreglo de dos dimensiones, el tamaño de una imagen se representa por la cantidad de píxeles en la dimensión llamada ancho y la cantidad de píxeles en la dimensión llamada largo. Las imágenes desde el punto de vista computacional, son representadas por una matriz de números enteros, en el rango de 0 a 255; cada valor de intensidad corresponde a un píxel en la imagen analógica con dimensiones iguales a las de la imagen [6]. Esta representación numérica de la imagen es transferida a un proceso de extracción de características (interpretación computacional de la imagen), donde eventualmente son representadas como un vector de características de la imagen; posteriormente, se pasan a un proceso de clasificación utilizando métodos de aprendizaje computacional o también redes neuronales que caen dentro del ámbito de la inteligencia artificial [7].

Una de las propuestas es el uso de un algoritmo computacional que ayude en la clasificación de imágenes tomadas desde gran altura y que resulte en la agrupación del total de imágenes en varias clases diferenciadas entre sí por sus características. Y también, de manera opuesta, todas las imágenes que corresponden a una misma clase, comparten características similares.

El presente documento describe el proceso y los resultados obtenidos de una serie de experimentos llevados a cabo con la finalidad de evaluar el desempeño de modelos de clasificación de imágenes remotas. Se trabaja con varios modelos de clasificación, a los cuales se les mide su desempeño empleando una serie de métricas para tal fin; de tal manera que podamos hacer comparaciones entre cada uno y resaltar sus principales ventajas. La clasificación se lleva a cabo sobre tres conjuntos de imágenes remotas, cada uno de ellos con características que lo diferencian de los otros y que ayudan a evaluar el desempeño de los clasificadores.

Conjuntos de imágenes remotas.

En su inicio, la clasificación de imágenes remotas era un trabajo manual y lento, debido a que no se contaba con conjuntos de imágenes remotas lo suficientemente grandes para ser alimentados a un algoritmo clasificador que realizara el trabajo automáticamente y sin sobre ajuste. En la actualidad ya se cuenta con bases de datos de imágenes remotas lo suficientemente grandes para trabajar con modelos de aprendizaje computacional. Para este trabajo empleamos los conjuntos de imágenes de la Universidad de California en Merced (UCM o UC Merced) [8]; el conjunto de imágenes tomadas a gran altura de la ciudad de Sydney Australia (Sydney) [9]; y el conjunto de datos de imágenes remotas y textos descriptivos (RSICD, por sus siglas en inglés) [10], construido por un conjunto de investigadores que trabajaban en la clasificación de imágenes remotas.

Todos estos conjuntos de datos contienen imágenes separadas en varias clases o grupos con distintas cantidades de imágenes remotas cada una. Pero dentro de cada clase, todas las imágenes son similares entre sí, de tal forma que todas son identificables como pertenecientes a esa clase.

Sobre la clasificación de imágenes remotas.

La clasificación de imágenes es un proceso de varios pasos, donde inicialmente seleccionamos un conjunto de imágenes remotas que son ajustadas para cumplir con los requerimientos del algoritmo clasificador; luego se extraen las características más sobresalientes de estas imágenes; después se lleva a cabo la clasificación de las imágenes caracterizadas y al final se ejecuta un proceso de evaluación del algoritmo para decidir si es necesario el cambio de algunos parámetros para mejorar su desempeño. El proceso es cíclico, iniciando de nuevo una vez que se establecen nuevos parámetros para realizar una nueva clasificación.

En este proyecto se utilizan técnicas de clasificación denominadas supervisadas, debido a que contamos con un conjunto de imágenes donde cada una de ellas ha sido previamente etiquetada con la clase a la que pertenece. Este proceso de etiquetado de imágenes, se lleva a cabo por humanos expertos en el área.

Cuando se dice clasificación de imágenes, nos referimos a que a cada imagen, se le asigna una y solo una clase o categoría como resultado del proceso de clasificación. A este método se le denomina *basado en píxeles*. Este proceso considera a toda la imagen en su totalidad y del resultado del análisis de la imagen completa, se le asigna una clase determinada a la que se predice que pertenece. No se lleva a cabo la identificación de objetos dentro de las imágenes.

A través de los años, han existido distintas metodologías para realizar la clasificación de imágenes. Con los últimos avances tanto en la captura de imágenes remotas como de los algoritmos de clasificación podemos utilizar imágenes de alta definición y algoritmos de clasificación con más uso en la

actualidad. Para este proyecto se han empleado métodos de clasificación basados en redes neuronales artificiales. Las redes neuronales empleadas son del tipo convolucionales que son las empleadas más comúnmente para el análisis e interpretación de características en imágenes digitales [11]. Los algoritmos de clasificación implementados se diferencian en la cantidad de capas empleadas (profundidad), y en los métodos que emplean para manejar el efecto de la desaparición del gradiente de pérdida y poder aumentar la profundidad de la red.

La manera tradicional de comparar a los diferentes métodos de clasificación es a través de la medición de la exactitud de sus predicciones; es decir, la cantidad de predicciones verdaderas del total de imágenes procesadas. Algunos de los conjuntos de imágenes remotas utilizados están desbalanceados porque algunas de sus clases contienen más imágenes que otras. Entonces, además de la métrica de exactitud, se emplean algunas otras que nos ayudan a encontrar una base de comparación más realista del desempeño de los clasificadores [12].

Después de dar una revisión rápida a los usos y retos que se presentan en la implementación de la clasificación de imágenes remotas, podemos plantear algunas de las problemáticas más predominantes.

Estos usos tan diversos en la clasificación de imágenes, presentan retos en cada campo de aplicación. Las imágenes tomadas desde gran altura presentan algunas problemáticas como la variación del punto de vista de la imagen porque no existe un punto de referencia para dicha imagen cuando se captura, de tal forma que se presentan problemas de oclusión, desorden de fondo y de iluminación [13], [14].

Otro reto presente es que actualmente los conjuntos de datos disponibles son reducidos; en varios estudios se elaboran bases de datos de imágenes específicamente para dicho estudio y no se encuentran públicamente. También es

difícil encontrar bases de datos ya etiquetadas con una extensa cantidad de imágenes para cada clase, de tal forma que facilite el proceso de clasificación [10].

Existen diferentes modelos de redes neuronales con aprendizaje profundo para la clasificación de imágenes remotas que presumiblemente tienen un desempeño distinto cada uno de ellos. Se piensa así, porque estos modelos de redes neuronales profundas son estructuralmente distintos entre ellos y en algunos casos surgen como respuesta para solucionar los problemas encontrados en modelos anteriores.

Evaluación del desempeño.

Por otra parte, también se han ido desarrollando a través del tiempo diferentes métricas de evaluación del desempeño de los mencionados modelos. Entonces, una de las líneas de acción del presente documento es presentar las configuraciones de parámetros, que resulten en los mejores valores de métricas de desempeño para cada uno de los modelos de clasificación de imágenes remotas bajo condiciones similares.

La ejecución de los modelos de redes neuronales profundas imponen una carga alta a los equipos de cómputo debido a la cantidad de operaciones necesarias para el entrenamiento de los datos [15]. Por lo tanto, se hace necesario el procesamiento suficiente para ejecutar la mayor cantidad de operaciones de computadora en el menor tiempo posible.

Como ya hemos mencionado, consideraremos varios conjuntos de datos de imágenes remotas, pero con condiciones distintas cada uno. Entonces estas condiciones presuponen desempeños distintos de los algoritmos al utilizar distintas bases de datos; es deseable conocer qué conjunto de datos desempeña mejor que otros e identificar las características que sustentan dicha mejora, para que una vez identificadas podamos obtener bases de datos similares que nos sirvan

de punto de partida para elaborar conjuntos de datos que cada vez tengan un incremento en su desempeño.

Objetivo general.

El objetivo general del proyecto es el siguiente:

Implementar y evaluar una serie de técnicas existentes usadas en la clasificación de imágenes remotas, particularmente aprendizaje profundo con redes neuronales artificiales, con el fin de conocer cuál de ellas tiene mejor desempeño en las métricas asociadas a la medición del proceso de clasificación cuando usamos diversas combinaciones de modelos y conjuntos de datos.

Los objetivos específicos nos ayudan a materializar el objetivo general del proyecto y nos proporcionan elementos entregables que podemos conocer de una manera más tangible.

Objetivos específicos.

Los objetivos específicos que se han propuesto al concluir este proyecto son:

1. Búsqueda y selección de conjuntos de datos de imágenes remotas apropiados para ser utilizados en las evaluaciones de los modelos de clasificación.
2. Búsqueda y selección de modelos de aprendizaje profundo con redes neuronales que posean características distintivas entre sí para llevar a cabo el proceso de clasificación de imágenes.
3. Obtener los parámetros de entrenamiento para cada modelo, mediante los cuales se obtenga el mejor valor de la métrica de exactitud.

4. Comparar el desempeño y los tiempos de ejecución de las arquitecturas empleadas en la clasificación de imágenes remotas, usando métricas predefinidas.

Descripción de la metodología general.

En el presente trabajo, se ponen en práctica algunos de los conocimientos adquiridos durante los cursos de la maestría en ciencia de datos e información. Se lleva a cabo la implementación real de algunos métodos de clasificación de imágenes remotas, empleando redes neuronales convolutorias.

Se experimenta con el ajuste controlado de los parámetros de los clasificadores para obtener la mejor medición de exactitud utilizando un conjunto de datos de entrenamiento. Posteriormente, este modelo ajustado, se vuelve a ejecutar practicando con un conjunto de datos distinto, llamado de validación; mediante el cual podemos medir la generalización del modelo.

Para cada combinación de modelo de clasificación y conjunto de imágenes se mide su desempeño y se llevan a cabo las comparaciones usando una tabla de resultados. Además de la medición del desempeño del modelo, se mide también el tiempo de ejecución en la etapa de clasificación usando dos tipos de procesadores, una computadora de escritorio común que usa CPU y por otra parte, una máquina que utiliza los más modernos procesadores en paralelo denominados GPU (unidades de procesamiento gráficas). La idea de la medición del tiempo de ejecución es conocer si estos algoritmos de clasificación necesitan máquinas especializadas, que a veces están fuera del alcance de los investigadores o, por el contrario, se pueden ejecutar en máquinas de acceso cotidiano.

En el capítulo 1 se detallan los fundamentos teóricos matemáticos de los diferentes modelos de clasificación de imágenes remotas, así como los conceptos

de redes neuronales artificiales y aprendizaje computacional en los que se basan dichos modelos. También en este capítulo, se describen los conjuntos de imágenes remotas utilizados, sus características y de donde se obtuvieron. Además, se explican las técnicas de medición que se utilizaron para la evaluación del desempeño de los modelos y conjuntos de datos.

En el capítulo 2 se presenta la metodología utilizada para la implementación de los modelos de clasificación empleados. Se muestra de qué manera se procesan los conjuntos de datos para que estén listos para su uso por parte del algoritmo de clasificación. Cómo se lleva a cabo el ajuste de los parámetros del modelo para encontrar el mejor desempeño. Y también que métricas de evaluación se implementan durante la implementación de los modelos de clasificación.

El capítulo 3 muestra diversas tablas de resultados de los experimentos llevados a cabo durante el proyecto, se muestran las predicciones logradas con los modelos de clasificación con su porcentaje de imágenes correctamente clasificadas para cada combinación de modelo y conjunto de datos. También se presentan los resultados de la evaluación del desempeño para cada combinación de modelo y conjunto de imágenes remotas.

Posteriormente, se documentan las conclusiones obtenidas de los experimentos realizados, que nos permiten resumir el conocimiento adquirido tras la finalización del proyecto.

Capítulo 1

Antecedentes

Capítulo 1. Antecedentes

1.1 Visión por computadora

El término visión por computadora se aplica a una rama de la Inteligencia Artificial (IA) que busca lograr la interpretación del medio ambiente, a través de imágenes digitales o videos. Dicha imagen es interpretada o “entendida” por un algoritmo computacional que ejecuta una serie de acciones predeterminadas, como podría ser la clasificación de dicha imagen o de los objetos dentro de ella [16].

Las máquinas tratan de llevar a cabo la función de visión, emulando el funcionamiento del ojo humano, pero lo hacen a través de cámaras digitales, imágenes, y algoritmos de cómputo.

La función de visión computacional se implementa a través del entrenamiento de máquinas y lo hace utilizando cámaras, datos y algoritmos en lugar de retina, nervios ópticos y corteza visual en el cerebro. Debido a que estos sistemas computacionales se entrenan para analizar miles de productos o procesos por minuto, notando imperfecciones que pasan desapercibidas para el ojo humano, podemos notar que rápidamente sobrepasan las habilidades de las personas [16].

Con la visión por computadora se describe e interpreta el medio ambiente que nos rodea mediante una o varias imágenes a través de la reconstrucción de sus propiedades, como la forma, iluminación y las distribuciones del color. La visión por computadora se emplea actualmente y como ejemplos se pueden mencionar el reconocimiento de códigos postales escritos a mano en cartas [17]; inspección de partes mecánicas para el control de calidad; construcción automática de modelos en 3D utilizando fotos aéreas para mapas; análisis de

imágenes médicas para el diagnóstico de enfermedades e inspección de tráfico en vialidades, entre otras [6].

1.2 Imágenes remotas.

En el presente documento nos referimos al término “imágenes remotas” (*Remote Sensing Images*, RSI, por sus siglas en inglés); como a aquellas imágenes de alta definición que son tomadas desde gran altura por satélites, aviones o vehículos no tripulados. Adicionalmente, se menciona que presentan características diferentes a las fotografías comunes tomadas en el día a día.

Estas imágenes remotas, que cubren una vasta área geográfica y que presentan una alta frecuencia temporal, ofrecen una buena oportunidad para obtener información del uso y cobertura del suelo a través de la interpretación de las mismas. La información frecuente y actualizada del uso del suelo es importante en varias actividades socioeconómicas como las aplicaciones en el cuidado del medio ambiente y la conservación de los recursos naturales [18].

La detección de objetos (por ejemplo automóviles, carreteras, edificios, etc.) en imágenes remotas sufre de varios retos que incluyen grandes alteraciones en la apariencia visual del objeto debido a una variación del punto de vista, oclusiones, iluminación, sombras, fondos borrosos, etc. Así como el crecimiento explosivo en cantidad y calidad que las nuevas áreas de aplicación requieren. Con los avances en la tecnología de sensores para imágenes remotas, últimamente se han podido obtener imágenes de satélites de alta resolución (VHR, por sus siglas en inglés) por ejemplo los satélites IKONOS, SPOT-5 y Quickbird, que han proporcionado información espacial y de textura aún más detallada. Por lo anterior, ahora se pueden reconocer una mayor cantidad de objetos hechos por el hombre debido a las resoluciones de menos de un metro que manejan los nuevos satélites [13]

1.3 Bases de datos de imágenes remotas.

Desde hace décadas, se han colocado cámaras fotográficas en satélites y aviones, lo que ha permitido capturar amplias regiones de la superficie terrestre con diferentes resoluciones. El avance tecnológico ha permitido un incremento en la resolución de las imágenes tomadas a distancia, así como también en la cantidad de canales de información que puede contener cada una de estas imágenes.

Con la finalidad de continuar con el estudio de la superficie terrestre y de los objetos que sobre ella se observan la comunidad científica ha procesado estas imágenes satelitales de alta definición y las ha transformado en colecciones de imágenes que se almacenan como bases de datos para su análisis y uso posterior.

En la presente investigación se explotaron tres conjuntos de datos de imágenes públicas que contienen la información necesaria para llevar a cabo los experimentos para el entrenamiento de modelos de clasificación de imágenes que son la parte esencial de la presente investigación.

Los conjuntos de imágenes se explican con más detalle a continuación.

1.3.1 Conjunto de datos UCM.

El conjunto de imágenes de uso de suelo de la Universidad de California en Merced, o simplemente conjunto de datos (dataset) UCM, es un conjunto de imágenes remotas que fue construido para el desarrollo experimental del documento [8] sobre clasificación de imágenes de uso del suelo.

Las imágenes de este dataset fueron etiquetadas manualmente con la clase a la que pertenece cada imagen. Consiste de 2,100 imágenes separadas en 21 clases sobre uso del suelo u objetos sobre la superficie, que han sido tomadas

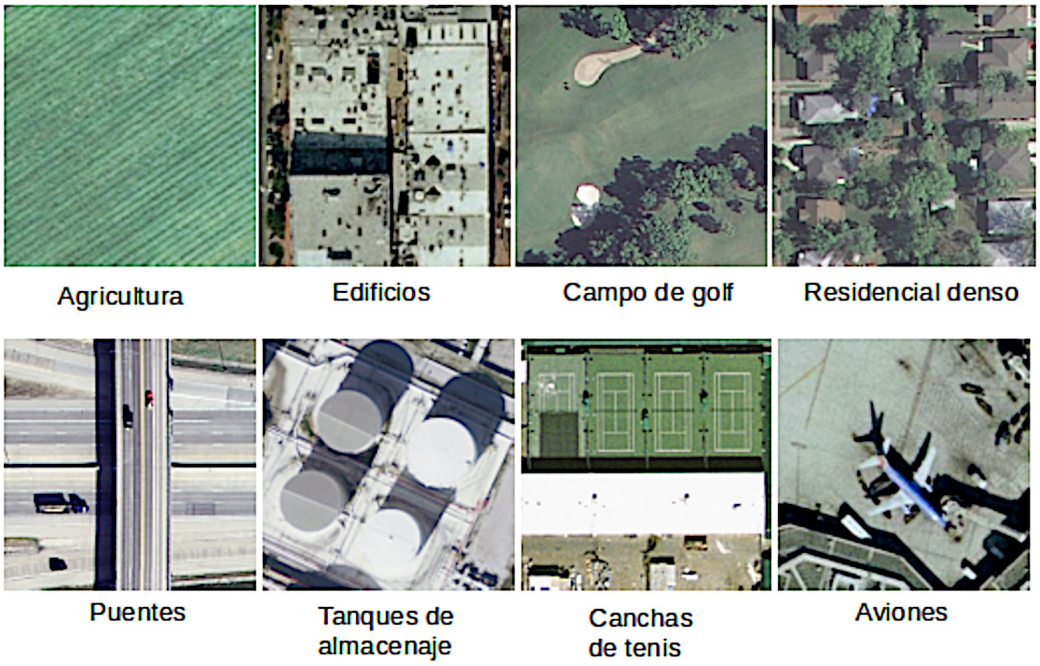


Figura 1: Ejemplos del conjunto de datos UCM. Fuente: [8].

desde el aire y pasado por un proceso de corrección llamado ortorrectificación que elimina los desplazamientos en la imagen y las variaciones de escala causadas por el relieve del terreno y la geometría del sensor. La resolución espacial es de 0.3048 m.

Las imágenes pertenecen a descargas de mapas del Servicio de Geología de los Estados Unidos de Norteamérica, de las siguientes regiones de los Estados Unidos: Birmingham, Boston, Buffalo, Columbus, Dallas, Harrisburg, Houston, Jacksonville, Las Vegas, Los Ángeles, Miami, Napa, New York, Reno, San Diego, Santa Bárbara, Seattle, Tampa, Tucson y Ventura. A cada una de las 21 clases se le asignaron 100 imágenes que miden 256×256 píxeles cada una. Las clases o categorías son: agricultura, aviones, campos de béisbol, playas, edificios,

chaparral, residencial denso, bosque, carretera, campo de golf, bahías, intersección vial, residencial con densidad media, campos de casas rodantes, puentes, estacionamientos, ríos, pistas de aterrizaje, residencial densidad baja, tanques de almacenamiento y canchas de tenis.

La figura 1 muestra ejemplos de imágenes del conjunto UC Merced. La base de datos completa se puede descargar de la página destinada para ello[1] en el servidor de la Universidad.

1.3.2 Conjunto de imágenes Sydney.

Este conjunto de datos fue construido en el trabajo descrito en [9] y extraída de una imagen satelital más grande (18,000 x 14,000 píxeles) de la ciudad de Sydney Australia obtenida desde Google Earth[2], la imagen tiene una resolución espacial de 0.5 m por píxel. Cada una de las imágenes de la base de datos tiene un tamaño de 500 x 500 píxeles, está digitalizada en formato TIFF que realiza la compresión de la imagen sin pérdida de información y cada una contiene tres capas de intensidad para cada uno de los colores primarios rojo, verde y azul (RGB, por sus siglas en inglés).

1 La base se puede descargar de http://weegee.vision.ucmerced.edu/datasets/landuse.html

2 La base de datos Sydney se puede obtener desde https://github.com/201528014227051/RSI-CD_optimal

Figura 2: Ejemplos de una imagen seleccionada de cada una de las 7 clases del conjunto Sydney . Fuente [9].

En [10] se menciona que este conjunto contiene 7 clases de imágenes que incluyen escenas de residenciales, aeropuertos, praderas, ríos, océanos, industrias y pistas de aterrizaje con un total de 613 imágenes. Adicionalmente, cada imagen contiene cinco textos descriptivos, pero la cantidad de imágenes por clase es variable como se muestra en la tabla 1. Esta base de datos contiene menos imágenes que la UCM descrita anteriormente. En la figura 2 se pueden observar ejemplos de imágenes de esta base de datos.

Clase	Número de imágenes
Residencial	242
Aeropuerto	22
Praderas	50
Ríos	45
Océanos	92
Industrial	96
Pistas de aterrizaje	66
Total	**613**

Tabla 1: Listado de las clases contenidas en la base de datos Sydney y la cantidad de imágenes en cada una. Fuente: Elaboración propia.

1.3.3 Conjunto de imágenes RSICD.

Este conjunto de imágenes se denomina “Base de datos de textos descriptivos en imágenes remotas” (RSICD, “*Remote Sensing Image Captioning*”, por sus siglas en inglés) y fue construido por Lu et al. en [10] para avanzar en la mejora del desempeño de textos descriptivos en imágenes remotas o RSIC[3].

Esta base de datos consta de 10,921 imágenes en total distribuidas en 30 clases, sin embargo, no todas las clases tienen la misma cantidad de imágenes, la tabla 2 muestra la distribución de imágenes por clase para RSICD. Las imágenes utilizan la compresión digital implementada por el grupo conjunto de expertos en fotografía (JPEG, por sus siglas en inglés). Este formato puede manejar colores de 24 bits, pero al momento de la compresión acepta una pérdida de información.

3 La base de datos RSICD se puede obtener desde https://github.com/201528014227051/RSICD_optimal

Cada imagen tiene unas dimensiones de 224 x 224 píxeles y la resolución espacial es variable. Las clases en las que se agruparon las imágenes son: aeropuertos, baldíos, campos de béisbol, playas, puentes, centros comerciales, iglesias, comercios, residencial denso, desierto, sembradíos, bosque, industrias, praderas, residencial medio, montaña, parques, escuelas, plazas, estacionamientos, área de juegos, estanques, viaductos, puertos, estación de tren, complejos, ríos, residencial escaso, tanque de almacenaje y estadios.

Este conjunto de imágenes remotas tiene la particularidad de ser uno de los más grandes reunidos a la fecha, adicionalmente cada una de sus clases no tiene la misma cantidad de imágenes, lo que le da una característica importante de desbalance entre ellas. La figura 3 nos muestra unos ejemplos de las imágenes que contiene el conjunto.

Figura 3: Ejemplos del conjunto RSICD . Fuente [10].

1.4 Clasificación de imágenes remotas.

Las imágenes captadas mediante sensores remotos en la forma de fotografía aérea han sido una fuente importante en la obtención de información sobre uso y cobertura del suelo. La clasificación de imágenes remotas es el proceso mediante el cual los píxeles de la imagen son agrupados de acuerdo a las similitudes de sus propiedades espectrales, si un pixel satisface ciertos criterios entonces es asignado a la clase correspondiente a dichos criterios. Se puede entender como la extracción de clases diferenciables o temas, a partir de imágenes remotas obtenidas desde satélites o naves no tripuladas [19].

La idea detrás de la clasificación de imágenes es que diferentes características de la superficie de la tierra tienen reflectancias diferentes. En el

contexto de las imágenes remotas, un píxel es el área de suelo que corresponde a una cantidad de intensidad dentro del conjunto de imágenes digitales [19], [20], [21].

Clase	Imágenes	Clase	Imágenes	Clase	Imágenes
Aeropuerto	420	Sembradíos	370	Área de Juegos	1031
Baldíos	310	Bosque	250	Estanque	420
Campo Base-ball	276	Industrias	390	Viaducto	420
Playa	400	Praderas	280	Puerto	389
Puente	459	Residencial Me-dio	290	Estación Tren	260
Centro Comer-cial	260	Montaña	340	Complejo	290
Iglesia	240	Parque	350	Río	410
Comercio	350	Escuela	300	Residencial Es-caso	300
Residencial Denso	410	Plaza	330	Tanque Alma-cenaje	396
Desierto	300	Estacionamiento	390	Estadios	290

Tabla 2: Cantidad de imágenes por clase en el conjunto RSICD. Fuente: [10].

1.5 El proceso de clasificación de imágenes remotas.

De acuerdo a [22], los pasos más importantes en el proceso de la clasificación de imágenes remotas son los siguientes, la figura 4 muestra un diagrama del proceso.

1.5.1 Selección del conjunto de imágenes remotas

Las imágenes obtenidas remotamente ya sean sensores colocados a escala del aire o a escala del espacio, varían en cuanto a su resolución espacial, radioeléctrica, espectral y temporal. Entonces, el encontrar un sensor de imágenes adecuado es el primer paso para una clasificación exitosa para el propósito deseado. También requiere la consideración de otros factores como las necesidades del usuario, la escala y características del área de estudio, la disponibilidad del conjunto de imágenes y las restricciones de costo y tiempo. La escala, la resolución de las imágenes y las necesidades del usuario son los factores más importantes en la selección de un conjunto de imágenes remotas [22].

1.5.2 Selección del sistema de clasificación y las muestras de entrenamiento.

Un sistema de clasificación adecuado, en conjunto con un número suficiente de imágenes remotas son los requisitos mínimos para lograr una clasificación satisfactoria [22].

En términos generales, un sistema de clasificación se puede integrar con base en la resolución espacial, las características del área de estudio, el tipo de conjunto de datos seleccionado y las necesidades del usuario. Es decir, considerar el objetivo que se quiere lograr y conseguir un conjunto de imágenes que vayan de acuerdo con el proyecto.

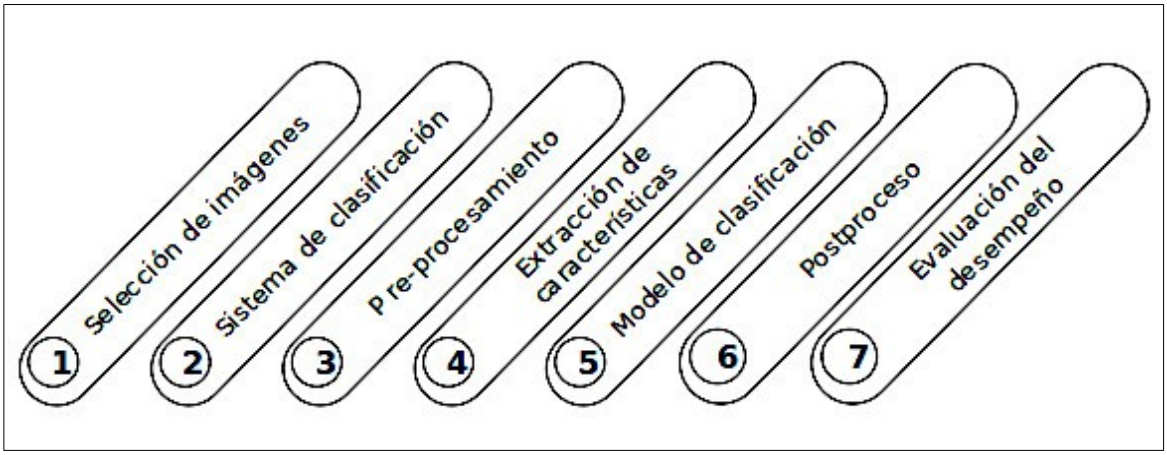


Figura 4: Proceso general de clasificación de imágenes. Fuente: Elaboración propia.

Un número suficiente de muestras de entrenamiento y prueba son necesarias para lograr la representatividad necesaria por clase. Estas muestras de entrenamiento son recolectadas de imágenes de satélites y mediante fotografías aéreas de gran resolución [23].

1.5.3 Preprocesamiento de imágenes.

Durante el flujo del proceso de clasificación se presentan dos momentos en los que se llevan a cabo preprocesamiento de las imágenes a analizar.

El primer momento sucede antes de la formación del conjunto de entrenamiento y prueba y se refiere a la corrección realizada en las imágenes mismas para lograr uniformidad entre las imágenes sometidas al análisis. El preprocesamiento de la imagen comprende entre otros mecanismos, detección y corrección de píxeles dañados, la rectificación geométrica y el etiquetado original por personas de las imágenes dentro de cada clase.

El segundo momento para el preprocesamiento es cuando estamos llevando a cabo el entrenamiento del modelo de clasificación. Este mecanismo se

utiliza sobre todo cuando la cantidad de imágenes es reducida. Se toma una muestra aleatoria de imágenes denominada lote y se someten a modificaciones de variación de ángulo, deslizamiento, rotación, giros de 180 grados, entre otros. De esta forma inyectamos al modelo nuevas imágenes que son una modificación de las existentes, lo que ocasiona un modelo más robusto [24].

1.5.4 Extracción y selección de características.

La extracción de características de las imágenes es uno de los pasos más importantes en el proceso de clasificación. Se encarga de llevar a cabo una interpretación numérica de las características físicas de las imágenes y adaptarlas al modelo de clasificación de tal forma que el proceso sea más eficiente.

Como se detalla en [7], las características más relevantes durante el proceso de clasificación son las siguientes.

La característica del color es tal vez la más importante. El histograma de colores es el método más común para extraer las características del color. La eficacia de la característica del color es independiente e insensible al tamaño, rotación y acercamientos en la imagen [25].

Las características de extracción de texturas es una técnica muy robusta cuando la imagen contiene una región repetitiva. Se dividen en dos tipos, texturas espaciales y espectrales [26].

Las técnicas de extracción de características basadas en la forma son usadas en el reconocimiento de objetos. Se dividen en basados en región o en el contorno. Este último método calcula las características tomando en cuenta los bordes, mientras que el método de región se basa en toda la superficie. [27]

Debido a la alta dimensionalidad de las características de las imágenes, se vuelve necesario seleccionar sólo aquellas variables que son más representativas

para el experimento. En la reducción de dimensiones, se han utilizado varios enfoques como análisis de componentes principales (PCA, por sus siglas en inglés), análisis de discriminantes y la extracción de características ponderadas no paramétricas, entre otros [28].

1.5.5 Selección del método de clasificación.

Algunos factores influyen en la selección del modelo de clasificación, tales como la resolución espacial y espectral de las imágenes, la disponibilidad de los conjuntos de imágenes a analizar y también el software del que disponemos para realizar el análisis. Dependiendo del modelo de clasificación seleccionado, se pueden obtener distintos resultados cuando se analiza el mismo conjunto de imágenes. En el capítulo 3 se detallan algunos de los resultados de los modelos de clasificación usados.

1.5.6 Procesamiento posterior a la clasificación.

Los clasificadores tradicionales basados en el análisis de píxeles, tienden a crear ciertas distorsiones o ruido después de la etapa de clasificación. Entonces se hace necesaria la aplicación de medidas correctivas, a través de filtros, para reducir dicho ruido. Por otra parte, debido a la complejidad del medio ambiente, es necesario recurrir a datos auxiliares para modificar la imagen clasificada basados en reglas establecidas por expertos. [29], [30]

1.5.7 Evaluación del desempeño del modelo de clasificación.

Evaluar el desempeño se entiende como una medida de referencia que se utiliza para comparar la calidad de clasificación del modelo empleado. En este trabajo se utilizan las siguientes medidas de desempeño, exactitud, precisión, recall y f1-score. De esta manera tenemos varios puntos de vista del desempeño y podemos tener una visión más completa de la calidad de cada modelo de clasificación y

compararlos entre sí. Los métodos de evaluación del desempeño empleados en el presente trabajo se detallarán más adelante.

1.6 Trabajos relacionados.

La clasificación automática de imágenes remotas se ha convertido en un tema importante de investigación en años recientes. Es uno de los tópicos dentro de la visión por computadora y forma parte de las bases del reconocimiento visual [31].

La mejora en el desempeño de los métodos de clasificación de imágenes remotas tienden a ampliar sus campos de aplicación [32]. Por ejemplo, uno de los usos más empleados es el de uso y cobertura del suelo; donde podemos observar a través del tiempo, el crecimiento de grandes áreas de terreno destinadas al uso humano, el desarrollo de la tierra cultivable y las áreas boscosas o desérticas [33], [34], [35]. Además, mediante la clasificación de imágenes remotas podemos localizar volúmenes de agua [30], [33], [34], [36]; identificar distintos tipos de vegetación [5], [36], [37], [38], así como la identificación de algunos contaminantes en el medio ambiente como O_3, $PM_{2.5}$, y PM_{10} [4], [39].

Para el proceso de predicción de imágenes en categorías existen métodos de clasificación que se han desarrollado con el tiempo. Podemos separar estos métodos en clásicos y avanzados. El desempeño de los métodos clásicos depende de los parámetros seleccionados para el modelo y de la cantidad de variables involucradas. Los métodos de clasificación avanzados han superado en el desempeño a los modelos clásicos debido a la habilidad para manejar una mayor cantidad de datos y su mayor poder de procesamiento [21].

Entre los métodos de clasificadores clásicos tenemos a *K-Means* e ISODATA, que son métodos no supervisados. El método de *K-Means* usa K grupos seleccionados aleatoriamente, cada píxel seleccionado inicialmente se

designa como el centroide del grupo, se calcula cada distancia desde cada píxel al centroide del grupo y se reasigna cada píxel a su centroide más cercano. Este proceso se repite hasta que los píxeles no cambien de grupo [38].

El método de clasificación ISODATA, es una versión alternativa de *K-Means*, donde se tiene una mejora en la asignación de los píxeles a cada grupo y no se reajustan los centroides constantemente, otra característica importante es que la cantidad de grupos se calcula automáticamente por el algoritmo [40], [41].

Los árboles de decisión son un grupo de métodos de clasificación no supervisados y no paramétricos que se han utilizado para el análisis de imágenes remotas sobre áreas cultivadas de maíz y forrajes [42]. Se han empleado para el análisis del uso del suelo utilizando imágenes multiespectrales [43], [44].

Dentro de los métodos de clasificación denominados avanzados, se contemplan las técnicas de máquinas de soporte vectorial (SVM, por sus siglas en inglés), los clasificadores de conjunto (*ensemble classifiers*) y las redes neuronales artificiales (ANN, por sus siglas en inglés) [21], [45].

Las máquinas de soporte vectorial en su forma más básica, son clasificadores binarios lineales que identifican un umbral entre dos clases; y que emplean optimización cuadrática convexa para obtener una solución óptima global [35], [46], [47]. Se han usado en la predicción de áreas de plantíos de palmares [37]; así como en la clasificación de distintas estructuras a nivel del suelo a través de imágenes hiperespectrales [48].

Los clasificadores de conjunto, se han implementado con la finalidad de incrementar el desempeño de los clasificadores individuales. Manejan tres enfoques, una combinación secuencial de clasificadores, una combinación en paralelo y técnicas de manipulación sobre las muestras de entrenamiento; enfocados todos en la identificación del uso del suelo [49], [50], [51].

Las redes neuronales artificiales (ANN, por sus siglas en inglés) son las más utilizadas en la actualidad cuando nos enfrentamos a la tarea de clasificar imágenes en general e imágenes remotas en particular. En especial se usan las redes convolucionales (CNN por sus siglas en inglés) profundas cuando los problemas de clasificación se relacionan con imágenes [31], [52], [53]. Las redes CNN han logrado los primeros lugares en las competencias de clasificación de imágenes a gran escala, presentan una habilidad superior en el aprendizaje de las características de la imagen; también a diferencia de otros métodos, las CNN representan un proceso de aprendizaje autocontenido [52], [53], [54], [55], [56], [57], [58], [59], [60].

El uso efectivo de las características o atributos de las imágenes remotas como datos de entrada mejoran la exactitud de clasificación, así como también es necesaria la selección de los mapas de características más útiles para reducir la dimensionalidad de las variables de entrada [21]. El histograma de gradientes orientados (HOG, por sus siglas en inglés) [61], nos permite obtener una representación de la imagen calculando diferencias entre grupos de píxeles de una imagen [37]. Se ha empleado también en la detección de minas terrestres subterráneas [62] y en la detección de enfermedades de la planta del maíz en imágenes remotas [63].

La técnica de patrones binarios locales (LBP, por sus siglas en inglés) fue descrita por Ojala et al. en [64], y es otra técnica para la extracción de características de las imágenes previo al proceso de clasificación de las mismas donde se obtiene un conjunto de patrones uniformes que corresponden a características primitivas como bordes, esquinas y regiones en la imagen; es invariante a la rotación y a la variación de la escala de grises. Esta técnica se ha implementado en la clasificación de escenas en imágenes remotas [65]; en la clasificación de imágenes remotas de alta resolución de la vegetación en

humedales costeros de algunas regiones de China [66]; y la identificación de maleza con imágenes remotas [67].

La técnica de transformación de características de escala invariante (SIFT, por sus siglas en inglés), permite identificar regiones en la imagen en el dominio de la escala que son invariantes a la rotación, traslación y escala de la imagen y que son mínimamente afectadas por el ruido y pequeñas distorsiones [68]. Esta técnica se usa para corrección y empate de imágenes que han sido tomadas desde diferentes fuentes y en distintos periodos de tiempo [69], [70].

Posteriormente, los trabajos de Le Cun et al. [71] permitieron el reconocimiento de conjuntos de dígitos escritos a mano a través de redes con retropropagación. Las redes convolucionales aseguran en cierta medida la invarianza en desplazamiento, distorsión y escala. Aprovechan campos receptivos locales para extraer características visuales elementales como bordes, esquinas y puntos finales. Estas características son combinadas en capas posteriores para detectar otras características de más alto orden [58].

Las redes neuronales convolucionales han demostrado los mejores resultados en la clasificación de imágenes [52], [53]. Se emplean principalmente en la clasificación de uso del suelo [54], [59], [72], [73], [74]; otros autores las implementan para la localización de objetos sobre la superficie [75], [76], [77], [78]; o para la detección de zonas con vegetación [67], [79], [80].

Un número suficiente de imágenes de entrenamiento que representen el propósito del proyecto es un factor crítico en la clasificación de imágenes [23], [81].

Los avances en la clasificación de imágenes remotas ha hecho posible la creación de diversos tipos de conjuntos de imágenes que funcionan como insumos para dichos proyectos; a manera de agrupamiento tenemos los conjuntos de

imágenes de alta resolución, por ejemplo UCM, *Sydney*, *Brazilian Coffee Scenes,* NWPU VHR-10, WHU-RS, RSSCN7, NWPU-RESISC45 y RSICD [8], [9], [13], [56], [59], [76], [82], [83], [84], [85], que describen diferentes escenas de objetos sobre la superficie terrestre.

Existen conjuntos de imágenes de entrenamiento con más de tres capas de información, denominados multiespectrales, por ejemplo CWCS y *Hyperspectral Weed Dataset* [67], [72], [86], [87]. Y otros conjuntos de imágenes con decenas de capas de información, denominados hiperespectrales, como *Indian Pines Dataset* y GEOBIA [86].

Para medir el desempeño del modelo de clasificación, se usan varias técnicas como la exactitud, que nos muestra la proporción de clasificaciones correctas del total de muestras [29], [42], [88], [89], [90]; se emplean las métricas de precisión y recall cuando tenemos conjuntos de datos con clases desbalanceadas [7], [63], [75], [91]; otras métricas para la medición del desempeño de clasificación son el área bajo la curva (AUC, por sus siglas en inglés) y el coeficiente Kappa [12], [50], [88], [92].

1.7 Métodos para extracción de características en imágenes.

1.7.1 Métodos tradicionales.

La tarea principal de la clasificación de imágenes remotas es la asignación de una o varias etiquetas a cada vector de píxeles contenido en un cubo de datos de una imagen remota de alta resolución; sacados de las propiedades espectrales o espacio – espectrales de la imagen.

La clasificación de imágenes remotas que usan métodos tradicionales de aprendizaje computacional se basan en la obtención manual de las características de las imágenes para el entrenamiento de los clasificadores. Estos métodos

normalmente confían en la utilización de habilidades de la ingeniería y el conocimiento de expertos para el diseño de estas características que representan a las imágenes digitales, como por ejemplo, forma, textura, color, detalles espaciales y espectrales [93].

De acuerdo a Mehmood en [45] existen tres tipos de clasificación de imágenes remotas que son, la clasificación basada en píxeles, la basada en objetos y la basada en escenas. Y los últimos avances apuntan hacia el desarrollo de la clasificación basada en escenas, debido a las mejoras en los sensores de imágenes que otorgan más información y mayor resolución.

Algunos de los métodos manuales tradicionales comúnmente empleados incluyen a los histogramas de color (CH, por sus siglas en inglés), patrones binarios locales (LBP, por sus siglas en inglés), histograma de gradientes orientados (HOG, por sus siglas en inglés), transformación global de escala invariante de imagen (GIST, por sus siglas en inglés) y la transformación de características de escala invariante (SIFT, por sus siglas en inglés).

Los histogramas de color son sencillos y efectivos en la representación de características. La información del color en una imagen puede ser representada ya sea en un solo histograma de tres dimensiones o en tres histogramas de una dimensión. Estos esquemas de representación son invariantes a la rotación y la traslación de la imagen de entrada. Los histogramas de color son más rápidos cuando los comparamos con los atributos de forma y textura, durante la extracción de características [94].

El histograma de gradientes orientados (HOG) fue construido por Dalal y Triggs [61] para la detección de personas en imágenes. Se basa en la idea de que los objetos en ciertas áreas de la imagen pueden ser muy bien caracterizados por una distribución de gradientes de intensidad local o dirección de los bordes. Se

implementa dividiendo la imagen en pequeñas regiones espaciales, denominadas celdas. Cada celda acumula un histograma de orientaciones de borde sobre los píxeles de esa celda. Posteriormente, se normalizan varias celdas y se unen en grupos llamados bloques. A estos bloques normalizados se les llama histogramas de gradientes orientados.

El método de extracción de características llamado patrones binarios locales (LBP), describe los patrones de textura alrededor de un píxel central tomando las diferencias de intensidad entre los píxeles circundantes. El método se aplica en imágenes monocromáticas (escala de grises) y tiene como particularidad ser invariante a la rotación y cambios en los niveles de la escala de grises.

Con la aplicación de este mecanismo, se identifican ciertas áreas de la imagen que se denominan uniformes, el término uniforme se refiere a que existen un número limitado de transiciones o discontinuidades en la representación circular del patrón obtenido. Los patrones uniformes más frecuentes corresponden a bordes, esquinas y pequeñas regiones, que son identificadas como características que sirven en la detección de objetos en la imagen. [64]

En el reconocimiento de objetos en imágenes, se requiere de características locales que no sean afectadas por el desorden o la oclusión parcial. Un reconocimiento efectivo se puede lograr usando descriptores locales de la imagen muestreados de una cantidad mayor de áreas repetibles. El método de transformación de características de escala invariante (SIFT) [68], transforma una imagen en una colección grande de vectores de características locales, cada uno de los cuales es invariante a traslaciones, escalamientos y rotación de la imagen, y son parcialmente invariantes a cambios en la iluminación y las proyecciones en 3D.

Las características invariantes a la escala se identifican empleando un enfoque de filtrado por etapas. La primera etapa identifica localidades clave en escala y espacio, buscando localidades que son un mínimo o un máximo de una función de diferencia de Gaussianos[4]. Cada punto se usa para generar un vector de características que describe la región local muestreada relativa a su marco de coordenadas espacio-escalar. Este vector resultante de características se denomina llaves SIFT. Posteriormente, las llaves SIFT obtenidas de una imagen, son empleadas en un enfoque por indexado de vecinos cercanos, para identificar modelos de objetos candidatos.

La transformación global de escala invariante de imagen (GIST) es un modelo computacional para el reconocimiento de categorías de escenas, que estima la forma de una escena utilizando unas pocas dimensiones perceptibles dedicadas para describir las propiedades espaciales de la escena. A estas propiedades espaciales de la escena se les denomina propiedades del envolvente espacial. Dichas propiedades proporcionan una descripción significativa de la escena en la imagen.

Una escena comúnmente se entiende como un arreglo de objetos sin restricciones y, por lo tanto, su reconocimiento semántico pasa por encontrar los objetos y su lugar exacto en la escena. GIST expone que, en vez de considerar una escena como una configuración de objetos, se le debe considerar como un objeto individual, con forma unitaria. Las propiedades del envolvente espacial son: el grado de naturalidad, grado de apertura, de rugosidad, de expansión y de robustez. [95]

4 La diferencia de Gaussianos es un algoritmo para el mejoramiento de las características de una imagen. Puede ser utilizado para incrementar la visibilidad de los bordes u otros detalles en las imágenes digitales.

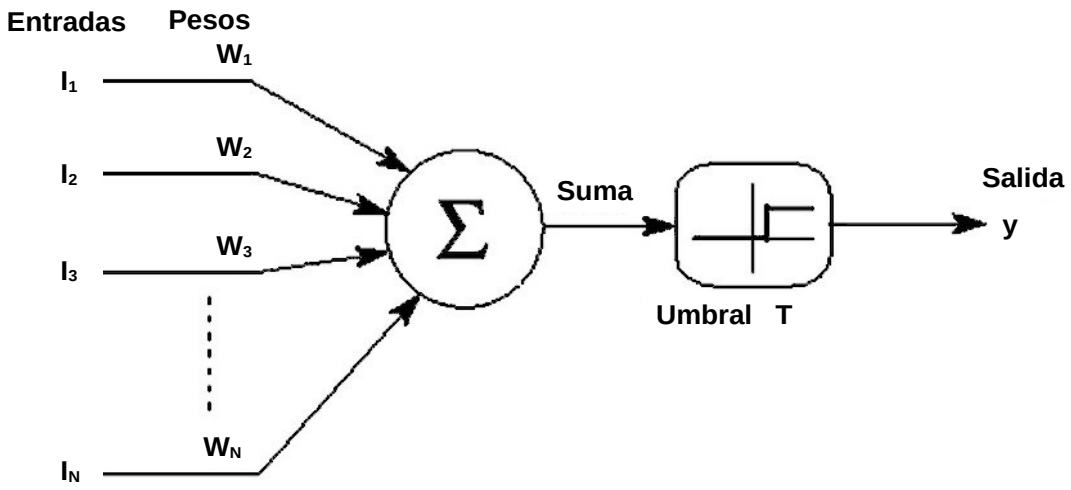


Figura 5: Modelo de una neurona artificial por McCulloch y Pitts. Fuente [96].

1.7.2 Métodos basados en redes neuronales artificiales.

Los métodos de extracción de características explicados en el apartado anterior pueden representar efectivamente los atributos de una imagen y entonces funcionan bien para el conjunto de datos en el que se trabaja. Sin embargo, estos métodos carecen de robustez como para ser utilizados en otro conjunto de datos diferente, lo que vuelve poco efectiva su generalización. Además, la intervención humana en el desarrollo de esas características, afecta considerablemente el proceso de clasificación, ya que se requiere de un alto conocimiento de la materia cuando se diseñan estas características manuales [93].

1.8 Redes neuronales artificiales

A mediados del siglo XX, los científicos entendían la interconexión entre las reglas lógicas y el razonamiento como algo conectado directamente. La rama de la ciencia denominada inteligencia artificial se preguntaba si se podría personificar el pensamiento a través de una máquina con reglas lógicas. O desde otro punto de vista: ¿Se podría modelar el pensamiento como un proceso mental humano con

algunas reglas lógicas?. Y aquí es donde comienza la historia de las redes neuronales artificiales, a través del artículo original de Warren McCulloch y Walter Pitts [96]. Ellos propusieron la unidad de umbral binario como un modelo computacional para una neurona artificial, ver figura 5. Esta neurona matemática calcula una suma ponderada de sus n señales de entrada $X_j,\ j = 1, 2, \ldots, n$ y genera una salida de valor 1 si la suma es arriba de cierto umbral u. La respuesta es 0 en cualquier otro caso. La ecuación 1 lo expresa matemáticamente:

$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right) \tag{1}$$

Donde $\theta(\cdot)$ es una función escalón unitario y w_j es el peso sináptico asociado con la *j*-ésima entrada. Los pesos positivos corresponden a sinapsis excitatorias, mientras que los pesos negativos modelan sinapsis inhibitorias.

McCulloch y Pitts probaron en principio, que mediante pesos escogidos cuidadosamente, un arreglo de tales neuronas podrían realizar cálculos universales.

1.8.1 El Perceptrón.

La arquitectura de la neurona artificial de McCulloch y Pitts carece de varias características de las neuronas biológicas, como: patrones complejos de conectividad, procesamiento de valores continuos, y sobre todo un procedimiento de aprendizaje.

Tomando en cuenta estas limitaciones, Frank Rosenblatt introdujo en 1958 el llamado concepto de perceptrón. Según Rosenblatt [97], [98], un perceptrón es una red de transmisión de señales que contiene tres tipos de unidades generadoras de señales: las unidades sensoriales, las unidades de asociación y las unidades de respuesta.

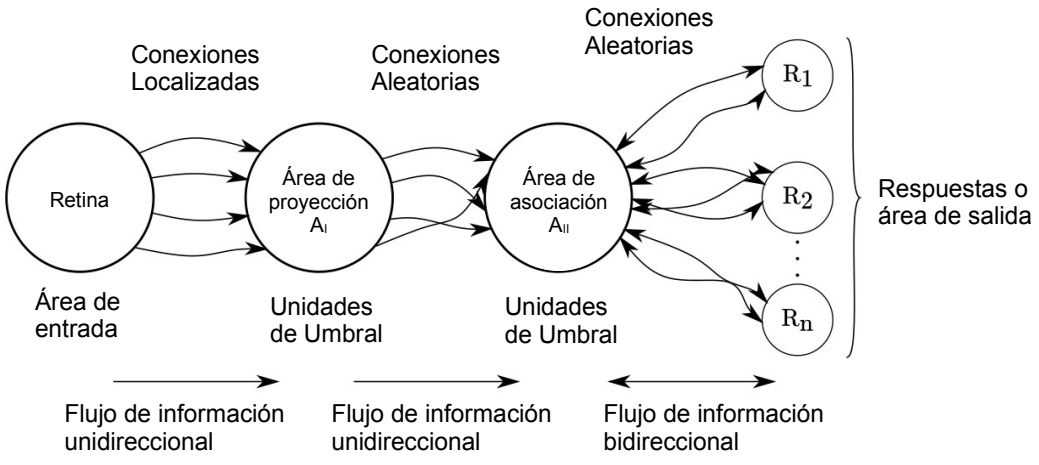


Figura 6: El foto-perceptrón de Rosenblatt. Fuente: [97].

El primer ejemplo proporcionado por Rosenblatt fue el llamado “foto-perceptrón” (ver Fig. 6), que intentaba emular la funcionalidad del ojo. Posteriormente, realizó mejoras a la arquitectura de su modelo mediante la introducción de un procedimiento de aprendizaje.

El principio básico de la regla de aprendizaje por corrección del error es utilizar la señal de error $(\hat{y} - y)$ para modificar los pesos de conexión y gradualmente reducir este error. La regla de aprendizaje usada en el perceptrón está basada en este principio de corrección del error.

La manera en la que el perceptrón logra la meta de aprendizaje es mediante el ajuste iterativo de los valores de sus pesos (enlaces de conexión entre unidades), hasta que encuentra un conjunto de pesos que permiten la separación de las clases del conjunto de datos.

Matemáticamente, el perceptrón puede ser descrito como [97]:

1. Una función lineal que agrega (suma) las señales de entrada

2. Una función de umbral que determina la respuesta de la neurona se activa o no.
3. Un procedimiento de aprendizaje para ajustar los pesos de conexión.

La función lineal que agrega las señales de entrada para una sola neurona, se define en la ecuación 2 cómo [99]:

$$z = b + \sum_{i=1}^{n} w_i x_i = b + w_1 x_1 + ... + w_n x_n \tag{2}$$

El peso para b puede ser aprendido de la misma forma que lo hacen los pesos de los valores de entrada.

La función del umbral de decisión se define en la ecuación 3:

$$\mathrm{y} = \mathrm{f(z)} = \begin{cases} +1, & \text{if } z > 0 \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

Donde z es la salida de la función de agregación lineal.

El procedimiento de aprendizaje se definió como un procedimiento de refuerzo por corrección de error. Este procedimiento garantizaba encontrar un conjunto de pesos que generara la respuesta correcta a cada problema, siempre y cuando tal conjunto de pesos exista [98]. La limitante a esta conclusión es que para muchos de los problemas relacionados con la ciencia cognitiva, este conjunto de pesos no existe. Cuando el problema de clasificación consiste de un conjunto de entrenamiento linealmente separable, los pesos si existen; en cualquier otro caso, no.

1. Calcular la diferencia entre el valor obtenido y el valor esperado para el conjunto de datos de entrenamiento.

2. Si los valores obtenido y esperado, son iguales, no hacer nada

3. Si los valores no son iguales, entonces calcular la diferencia (delta) entre estos dos valores.

4. Utilizar una porción de la diferencia para actualizar los pesos de la red.

La ecuación 4 describe el procedimiento anterior como [100]:

$$w_{k+1} = w_k + \eta(y - \hat{y})x_k \tag{4}$$

Donde:

w_k es el vector de pesos para el caso k

η es la razón de aprendizaje (entre 0 y 1)

y es el valor actual (valor verdadero)

$\hat{y}$ es el valor predicho (predicción)

x_k es el vector de entradas para el caso k

La razón de entrenamiento η tiene la función de facilitar el proceso de entrenamiento. Es decir, en lugar de reemplazar completamente los pesos anteriores con la suma de los pesos más un delta, únicamente se incorpora una proporción del error total en el proceso de actualización. Esto produce que el proceso de aprendizaje sea más estable en el tiempo.

1.8.2 Algoritmo de retropropagación (BP).

En los tiempos del perceptrón y su algoritmo de aprendizaje, no se tenía el conocimiento de cómo entrenar una red neuronal con más de una capa. En 1975, el economista Paul Werbos propuso la retropropagación (BP, por sus siglas en inglés), como una forma de propagar el error a través de la capa intermedia (oculta) cuando trabajaba en su tesis doctoral relacionada con la reducción del error en algoritmos predictivos dentro de las ciencias sociales. Pero este descubrimiento pasó desapercibido por varios años, hasta que fue redescubierto por David Parker en 1985. En ese mismo año, también Yann Le Cun propuso la retropropagación en investigaciones por separado [101]. Por último, la técnica de

retropropagación fue popularizada por Rumelhart, Hinton y Williams en [102] donde describen el algoritmo.

El algoritmo de BP está basado en la regla de aprendizaje de corrección de error, la cual usa una función para modificar los pesos de conexión y reducir gradualmente el error. Este error es la diferencia entre la salida obtenida por la red y un valor objetivo predefinido. El algoritmo de BP está considerado dentro del paradigma de aprendizaje supervisado porque se utiliza un valor objetivo existente. La técnica de descenso de gradiente es un enfoque común para minimizar los valores de una función; este principio se utiliza en la retropropagación para ajustar los pesos de conexión y así lograr la minimización de la función de error [103].

A continuación se muestra el algoritmo de retropropagación tomando de base el documento [102]. El procedimiento aplica para las redes que poseen una capa de entrada, cualquier número de capas intermedias y una capa de salida. Empezamos con la dirección hacia adelante hasta que calculamos una medida del costo o error de nuestra predicción en la capa de salida. Posteriormente, se inicia con el proceso de aprendizaje usando retropropagación (que a su vez se basa en descenso de gradiente), donde se lleva a cabo el ajuste de los pesos entre unidades para eventualmente obtener el menor valor posible del error.

Para cada unidad (neurona), tenemos una función agregadora z_j (ecuación 5) que representa una función lineal entre las entradas a esa unidad multiplicada por sus pesos.

$$z_j = \sum_i x_i w_{ji} \tag{5}$$

A las unidades se les pueden asignar sesgos, introduciendo una entrada extra a cada unidad que siempre tiene el valor de 1. El peso de esta entrada adicional, se denomina el sesgo y es tratada de manera similar a los otros pesos.

Cada unidad (neurona) tiene un valor de salida dentro de los números reales, y_j, que tiene la característica de ser una función no lineal del total de la entrada. En este caso en particular utilizaremos la función sigmoidea, ver ecuación 6 , que se usa comúnmente para este fin.

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{6}$$

Lo que se busca es un conjunto de pesos que aseguren que para cada vector de entrada, la red produce un vector de salida que es el mismo (o lo suficientemente cercano) al vector de salida objetivo. Contando con un número finito y fijo de casos de entrada – salida, se puede calcular el error total del desempeño de la red para ese particular conjunto de pesos; al obtener la diferencia entre el valor de salida obtenido por la red y el valor de salida objetivo.

El error total E, se define en la ecuación 7 como:

$$E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2 \tag{7}$$

Donde c es el índice sobre todos los casos (muestras de entrada y salida), j es el índice para identificar a las unidades (neuronas), y es la respuesta obtenida por la red como predicción en la capa de salida y d es la respuesta objetivo que tomamos como verdadera.

Para minimizar el error E por medio de descenso de gradiente es necesario calcular la derivada parcial de E con respecto a cada uno de los pesos en toda la red. Esto es simplemente la suma de las derivadas parciales de cada uno de los casos de entrada. Para un caso dado, la derivada parcial del error con respecto a

cada uno de los pesos, se calcula en dos fases. Ya se describió la fase hacia adelante cuando obtuvimos el error usando las funciones de agregación y de activación para cada neurona hasta la capa de salida. La fase hacia atrás o retropropagación que se encarga de transmitir las derivadas desde la capa de salida hacia la capa de entrada es un poco más compleja.

La fase hacia atrás empieza cuando calculamos $\partial E/\partial y$ para cada una de las unidades de salida, ver ecuación 8. Al diferenciar la ecuación del error E para un caso en particular c y suprimiendo ese índice, tenemos:

$$\frac{\partial E}{\partial y_j} = y_j - d_j \tag{8}$$

Siguiendo la dirección hacia atrás, empleando la regla de la cadena y diferenciando la ecuación de la función de activación sigmoidea, se obtiene $\partial E/\partial z_j$:

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \cdot y_j(1 - y_j) \tag{9}$$

La ecuación 9 nos permite conocer como un cambio en la función de agregación afecta el error. Pero esta función de agregación es una función lineal de los estados y los pesos de las unidades de la capa anterior, de tal forma que se puede calcular como se afecta el error al cambiar estos estados y pesos. Para un peso w_{ji}, la derivada del error con respecto a este peso se muestra en la ecuación 10:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} = \frac{\partial E}{\partial z_j} \cdot y_i \tag{10}$$

Ahora se puede calcular la derivada del error con respecto a la salida y_i de la capa oculta (o anterior). Tomando en cuenta todas las conexiones que salen de la unidad i, se obtiene la ecuación 11:

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{dz_j}{dy_i} \frac{\partial E}{\partial z_j} = \sum_j \frac{\partial E}{\partial z_j} \cdot w_{ji} \tag{11}$$

Los pasos que se presentaron desde $\partial E/\partial y_j$ hasta $\partial E/\partial y_i$ son el corazón del algoritmo de BP. Notemos que ahora se pueden repetir estos pasos para ir a través de tantas capas como se desee.

La fórmula utilizada para la actualización de los pesos entre las unidades, ver ecuación 12, es una versión muy simple del descenso de gradiente que cambia los pesos en una cantidad proporcional al total de la razón de cambio encontrada $\partial E/\partial w$:

$$\Delta w = -\varepsilon \partial E/\partial w \tag{12}$$

Este método es sencillo y puede ser fácilmente implementado con hardware que maneje procesamiento local en paralelo.

1.9 Autocodificadores apilados.

1.9.1 Autocodificadores.

Afortunadamente, debido a los avances realizados en el dominio del aprendizaje profundo con redes neuronales y también el incremento de conjuntos de datos de imágenes tomadas desde gran altura, es que nacen nuevos algoritmos que se pueden emplear en la extracción y reducción de características de las imágenes.

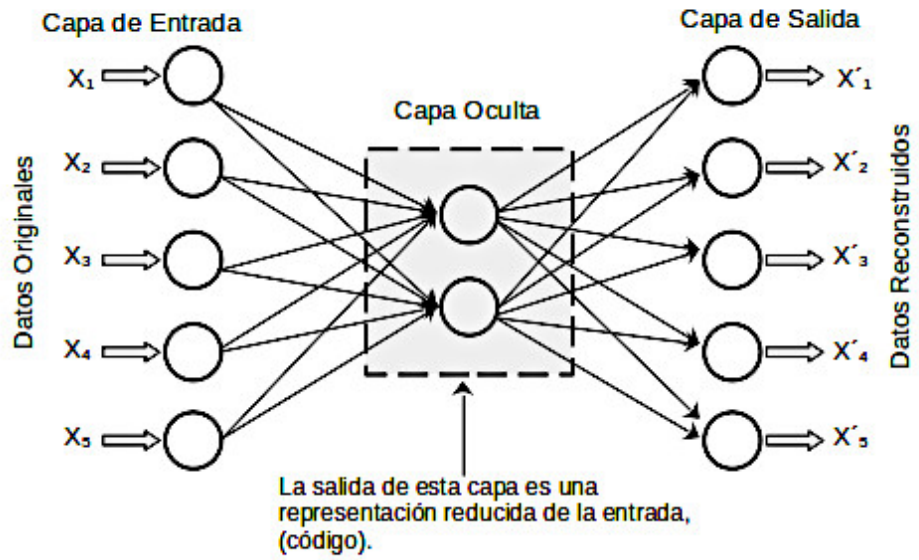


Figura 7: Autocodificador con una capa oculta de dos unidades. Fuente: [104].

Un autocodificador es una red neuronal que es entrenada para intentar copiar su entrada de datos en su salida. Internamente, tiene una capa oculta que contiene un código usado para representar a la entrada. Esta red neuronal puede ser vista como dos partes: una función de codificación y otra de decodificación que produce una reconstrucción de la entrada.

Si un autocodificador es exitoso en simplemente copiar su entrada a su salida, entonces no es de mucha utilidad. En lugar de eso, los autocodificadores se diseñan para que no realicen una copia exacta de la entrada; están restringidos de tal forma que hacen una copia aproximada y copiar aquellos datos que nos ayuden a reconstruir los datos de entrada.

Como el modelo se fuerza a priorizar qué aspectos de la entrada deben ser copiados, como consecuencia aprende propiedades útiles de los datos [104]. Esta reconstrucción imperfecta de los datos de entrada se logra colocando una cantidad menor de unidades (neuronas) en la capa oculta. Como resultado se obtiene una representación reducida de los datos, por lo tanto, a este tipo de reconstrucción se le denomina de pérdida. La función de pérdida en esta red neuronal utiliza la sumatoria de diferencias al cuadrado entre la entrada y la salida para forzar a la salida a ser tan similar a la entrada como sea posible. La representación general de un autocodificador se observa en la figura 7.

En un artículo del año 2006, G.E. Hinton y R.R. Salakhutdinov [105], propusieron una red neuronal para la reducción de dimensiones de un conjunto de datos de entrada. Describen una generalización no lineal del método de análisis de componentes principales (PCA, por sus siglas en inglés) que usa una red "codificadora" multicapa (*"encoder"*) que transforma la alta dimensionalidad de los datos en un "código" de pocas dimensiones y posteriormente una red "decodificadora" para recuperar los datos desde el código.

Comenzando con pesos aleatorios en las dos redes, se pueden entrenar juntas, minimizando la discrepancia entre los datos originales y su reconstrucción. Los gradientes requeridos se obtienen fácilmente utilizando la regla de la cadena para propagar hacia atrás las derivadas del error, primero a través de la red del decodificador y luego a través de la red del codificador. A todo este sistema se le llama autocodificador.

1.9.2 Autocodificadores apilados.

En las aplicaciones reales, se utilizan autocodificadores apilados multicapa (SAE, por sus siglas en inglés) para el aprendizaje de características de las imágenes. Las redes profundas multicapa, ofrecen un extraordinario poder de representación de los datos.

Esta idea de “apilamiento” se muestra en la figura 8 donde observamos el ejemplo tomado de [106]. Tenemos tres autocodificadores representados por las capas h_1, h_2 y h_3, que son apilados uno encima del otro para formar un autocodificador apilado. Cuando “encimamos” estos autocodificadores, la salida de la capa oculta del autocodificador previo, es la entrada del siguiente autocodificador.

Una de las actividades importantes en el entrenamiento de los autocodificadores apilados es cómo inicializamos la red. La forma en la que se inicializan los parámetros influencia la convergencia de la red, así como también, influye en la estabilidad del entrenamiento. G.E. Hinton y otros, en [105] encontraron una buena solución para la inicialización de los pesos de la red.

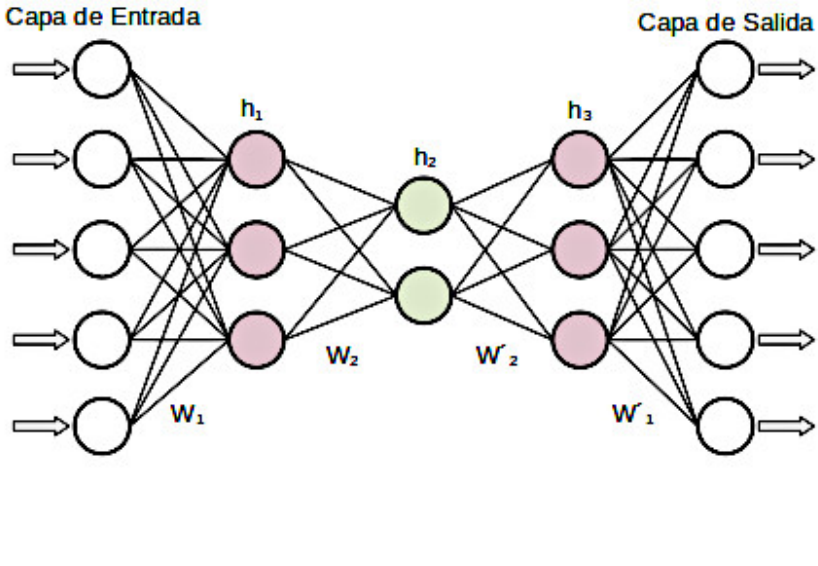


Figura 8: Autocodificador apilado con tres capas ocultas. Fuente: [104].

1.10 Máquinas restringidas Boltzmann (*Restricted Boltzmann machines*) y redes de creencia profunda (*deep belief networks*).

1.10.1 Máquinas Boltzmann restringidas.

Las máquinas restringidas Boltzmann (RBM, por sus siglas en inglés) son un modelo fundamentalmente diferente de las redes de alimentación hacia adelante. Las redes neuronales convencionales son redes que asignan un conjunto de entradas hacia otro conjunto de salidas. De manera distinta, las RBM son redes en las que los estados probabilísticos son aprendidos desde un conjunto de entrada, siendo esto útil en el modelado no supervisado [107].

Las RBM evolucionaron de un modelo clásico de red neuronal llamado red “Hopfield”. Este tipo de red se forma de nodos que contienen estados binarios, que a su vez representan valores de atributos binarios de los datos de entrenamiento.

La red Hopfield crea un modelo determinístico de las relaciones dentro de los diferentes atributos a través del uso de “pesos” entre los nodos de la red.

Eventualmente, la red Hopfield evolucionó en la que se conoce como máquina Boltzmann, la cual usa estados probabilísticos para representar las distribuciones Bernoulli de los atributos binarios. La máquina Boltzmann contiene tanto estados visibles como ocultos. Los estados visibles modelan las distribuciones de los datos observables, mientras que los estados ocultos modelan la distribución de las variables ocultas (latentes). La meta de esta máquina es aprender los parámetros del modelo de tal manera que la probabilidad se maximice [107].

Entonces retomando las RBM, podemos decir que estructuralmente, son solo máquinas Boltzmann que no tienen conexiones entre las neuronas de la misma capa (oculta a oculta y visible a visible). Esto parece un punto menor, pero es lo que hace posible el uso de una versión modificada de la técnica de retropropagación que se usa en las redes neuronales con alimentación hacia adelante [99]. Por lo anterior, las máquinas Boltzmann restringidas tienen dos capas, una visible y otra oculta. La capa visible es donde colocamos los datos de entrada y también leemos la información de salida.

Una RBM nos provee de una representación de la distribución de probabilidad que subyace de las observaciones. Puede ser utilizada para comparar las probabilidades de observaciones aún no vistas y para el muestreo desde una distribución aprendida. Por ejemplo, se puede corregir algunas observaciones parciales y obtener por muestreo las unidades visibles que falten para completar todas las observaciones [108].

1.10.2 Redes de creencia profunda (DBN).

Al modelo de apilamiento de varias máquinas Boltzmann restringidas utilizando un algoritmo “codicioso” de entrenamiento se le dio el nombre de red de creencia profunda (DBN por sus siglas en inglés) [109]. Las DBN permitieron que las redes neuronales fueran profundas mediante la inicialización de un buen conjunto de pesos (usando entrenamiento con RBM) para usarse en la etapa de retropropagación. Este buen punto de inicio para la optimización de la retropropagación ya no enfrentó el problema de la reducción del gradiente [110].

Como se mencionó antes, se puede entrenar una red neuronal usando RBM. Este entrenamiento puede resultar en una muy buena inicialización de los pesos para usar retropropagación en el entrenamiento de la red. Antes del desarrollo de la función de activación “Unidad lineal rectificada” (ReLU), y de las etapas de abandono (*“dropout”*), las redes de perceptrones multicapa no podían ser profundas por el problema del desvanecimiento del gradiente. Esto se debe a que los pesos iniciales aleatorios, no son lo suficientemente adecuados para empezar la optimización usando retropropagación, en especial en redes profundas. Por lo anterior, se propuso el método para preentrenar la red neuronal, lo cual inicializa la red a un adecuado conjunto de pesos; posteriormente los pesos sufren un ajuste fino usando la retropropagación [110].

Desde su aparición en 2006, la apatía presentada hacia las redes neuronales empezaba a terminar gradualmente, debido a que ahora las redes podrían ser profundas y posibilitaban el manejo de datos no lineales.

1.11 Redes neuronales convolucionales.

Las redes convolucionales nacen de la idea de lograr la visión por computadora, de la idea de detectar patrones con el ojo humano y estos mecanismos se trasladaron de una manera sencilla a la inteligencia artificial.

Las redes neuronales convolucionales están diseñadas para trabajar con entradas que manejan estructura de rejilla, las cuales tienen una fuerte dependencia espacial [111]. Desde luego el ejemplo más eminente de este tipo de datos son las imágenes en dos dimensiones. Se menciona también que se requiere una dimensión adicional para capturar los colores de la imagen, lo que crea un conjunto de datos en tres dimensiones. Una propiedad importante de las imágenes es que exhiben cierto nivel de invarianza a la traslación; por ejemplo, una naranja tiene la misma interpretación ya sea que se encuentre en la parte superior o inferior de la imagen.

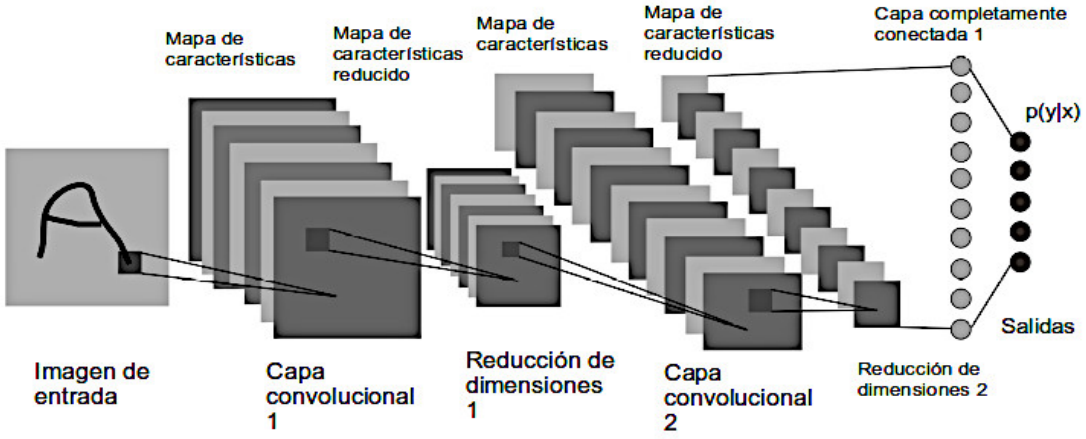


Figura 9: Estructura de una red neuronal convolucional. Fuente: [58].

El nombre de redes neuronales convolucionales (CNN) proviene del hecho que este tipo de redes manejan una operación matemática referida como “convolución”. Esta operación es el producto punto entre un conjunto de datos en dos dimensiones, comúnmente llamado “entradas” y otro conjunto de datos en dos dimensiones, denominado “filtro”, que se va moviendo a través de todo el conjunto de datos que crea la imagen de interés [107].

1.11.1 Capas Convolucionales.

Existen cuatro ideas principales detrás de las CNN, que toman ventaja de las propiedades de estas señales naturales: conexiones locales, pesos compartidos, combinación de características (*pooling*) y el uso de muchas capas.

La arquitectura de una CNN típica (ver Fig. 9), se arma como una serie de etapas. Las primeras etapas se componen de dos tipos de capas: las capas convolucionales y las capas de combinación. Las unidades en una capa convolucional son organizadas en mapas de características. Dentro de cada mapa, cada una de las unidades están conectadas a regiones locales en los mapas de características de capas anteriores a través de un conjunto de pesos llamados banco de filtros. El resultado de esta suma ponderada local, es pasada a través de una función no lineal como ReLU. Todas las unidades en un mapa de características comparten el mismo banco de filtros. Pero diferentes mapas de características en una capa, usan diferentes bancos de filtros [58].

Existen dos razones para dicha arquitectura. En primer lugar, en una matriz de datos, tal como las imágenes, los grupos locales de valores suelen estar altamente correlacionados, formando “motivos” locales distintivos que se detectan fácilmente. En segundo lugar, las estadísticas locales de imágenes no varían con la ubicación. En otras palabras, si un “motivo” puede aparecer en una parte de la imagen, podría aparecer en cualquier parte, de ahí la idea de unidades en diferentes lugares que comparten los mismos pesos y detectan el mismo patrón en diferentes partes de la imagen [111].

Al contar con campos receptivos locales, las neuronas pueden extraer características visuales elementales como bordes, terminaciones, esquinas o características similares. Estas características básicas son combinadas por las capas subsecuentes para detectar características de alto orden. Las unidades dentro de una capa están organizadas en planos dentro de los cuales todas las

unidades comparten los mismos pesos. A este conjunto de salidas de las unidades en dicho plano, se le llama mapa de características. Todas las unidades en un mapa de características están restringidas para ejecutar la misma operación en diferentes partes de la imagen.

Una capa convolucional completa está compuesta de varios mapas de características (con diferentes vectores de pesos), de tal manera que de una misma área espacial se puedan extraer múltiples características. Una implementación secuencial de un mapa de características recorrería la imagen de entrada con una sola unidad que tiene un campo receptivo local y almacenaría los estados de esta unidad en las ubicaciones correspondientes en el mapa de funciones. Esta operación es equivalente a una convolución, seguida de un sesgo (*bias*) y una función de reducción, de ahí el nombre de red convolucional [58].

Generalmente, el tamaño de los mapas de características de entrada es $H \cdot W \cdot C$ (alto H, ancho W y canales C). Cada *kernel* o filtro de convolución tiene dimensiones de $K \cdot K \cdot C$ y esto es porque el número de filtros debe ser igual al número de canales de entrada. Tomando la información que aparece en [31], las operaciones para el flujo de datos en la capa convolucional puede ser expresado a grandes rasgos según la ecuación 13:

$$FM_{salida} = f\left(\sum_{i=1}^{c} M_i * W_i + B\right) \tag{13}$$

Donde FM_{salida} representa el nuevo mapa de características resultante, c es la cantidad de filtros o *kernels*, M_i representa la matriz de entrada (imagen) en dos dimensiones, W_i es la matriz de pesos de los filtros de convolución, la matriz de sesgos es B y $f(\cdot)$ es la función de activación no lineal.

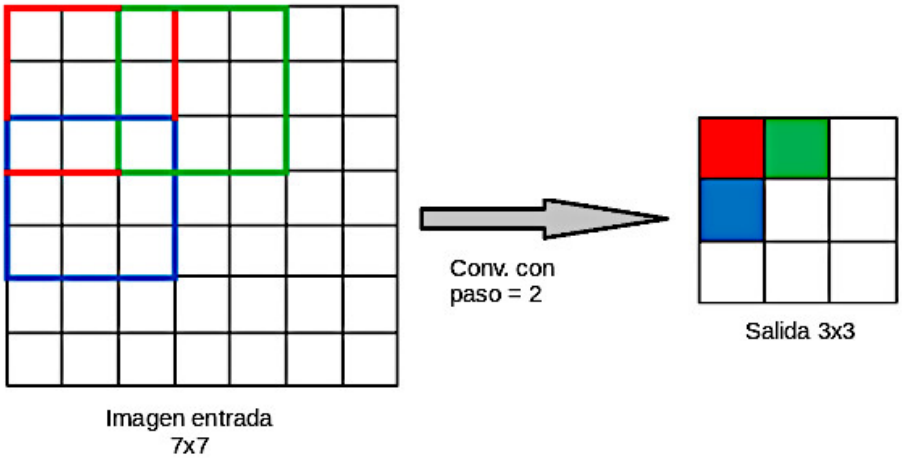


Figura 10: Ejemplo del concepto de *paso* (*stride*). Fuente: Elaboración propia.

Para conocer el tamaño de la matriz de características de salida o^2 como resultado del proceso de convolución, es necesario primero explicar el concepto de *relleno* (*padding*) que significa agregar píxeles alrededor de la imagen para completar las dimensiones de la matriz de entrada. Y explicar también el concepto de *paso* (*stride*) (ver Fig. 10), que nos ayuda a reducir el nivel de granularidad de la convolución, ya que nos permite mover el filtro o kernel en más de una unidad cuando se lleva a cabo el proceso de convolución. Normalmente, los valores para el *paso* son de 1 o 2 [107].

Entonces, para cualquier matriz de entrada de dos dimensiones de tamaño i^2, tamaño de filtro igual a k^2, *paso* igual a s y *relleno* igual a p, el tamaño del mapa de características de salida o^2 (ecuación 14) es:

$$o = \left[\frac{i + 2p - k}{s}\right] + 1 \tag{14}$$

El concepto de *relleno* se utiliza en las redes convolucionales, ya que cuando se realiza la operación de convolución, se reduce el tamaño de la matriz de la capa siguiente en comparación a la anterior. Este tipo de reducción de

dimensiones normalmente no es deseable porque se pierde algo de información de los bordes de la imagen o del mapa de características (en una capa oculta).

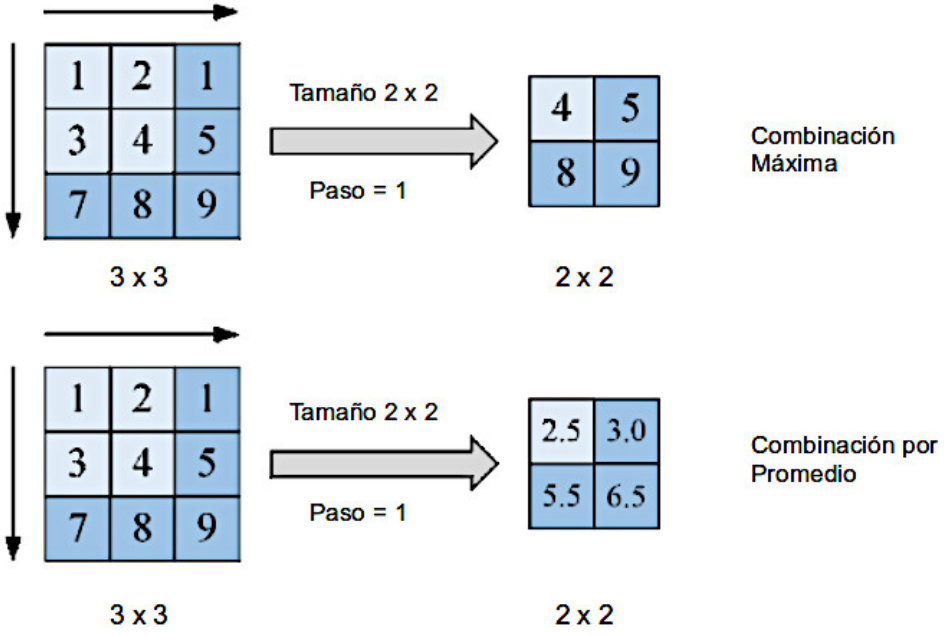


Figura 11: Método de combinación por promedios y máximos. Fuente: [31].

1.11.2 Capas de combinación.

Las capas de combinación (*pooling*) se colocan después de las capas convolutorias. La razón principal de utilizar estas capas de combinación es que llevan a cabo una reducción de las dimensiones de los datos de entrada para reducir el número de conexiones de las capas convolutorias y, por lo tanto, reducir la carga de procesamiento computacional [31].

También con esta capa de combinación no se modifican las principales características de la imagen de entrada al momento de ser reducida y además hace que el mapa de características de salida sea más robusto a la distorsión y al error generados por el proceso [112]. Los dos métodos más utilizados para las

capas de combinación, son los métodos de combinación por promedios y combinación por el máximo. En la figura 11, se observa el proceso de reducción de dimensiones usando los métodos de promedios y máximos. Como se documenta en [31], la relación entre los tamaños de las matrices de entrada y salida durante la operación de combinación, satisface la relación mostrada en la ecuación 15.

$$o = \left[\frac{(i-k)}{s} + 1\right] \tag{15}$$

A diferencia de las operaciones convolutorias, la operación de combinación es realizada para cada uno de los mapas de características. Es decir, el proceso de combinación opera independientemente sobre cada mapa para producir otro mapa de características, pero con dimensiones reducidas. La operación de combinación no cambia el número de mapas de características.

1.11.3 Capas completamente conectadas.

Cada mapa de características en la última etapa convolucional, está conectado a cada unidad de la capa oculta en la primera capa completamente conectada. Esta capa funciona exactamente de la misma manera a como funciona una red neuronal con alimentación hacia adelante. En las redes neuronales convolucionales es común encontrar más de una capa de unidades completamente conectadas hacia el final de la red, con la finalidad de incrementar el poder de clasificación de dicha red convolucional. Ya que estas capas completamente conectadas (FC, por sus siglas en inglés) están densamente conectadas, la mayor cantidad de parámetros de la red convolucional residen en esta etapa FC [31].

Las capas convolucionales tienen un número más grande de activaciones, mientras que las capas completamente conectadas tienen un número mayor de

conexiones; las primeras incrementan el espacio de memoria utilizado, mientras que las últimas incrementan la cantidad de parámetros usados en los cálculos. La capa de salida en una red neuronal convolucional, se configura de acuerdo a la aplicación que se ocupe.

Para el caso de la clasificación de imágenes, la capa de salida está completamente conectada a cada unidad (neurona) de la penúltima capa y tiene los pesos asociados a ellas. Entonces, se puede usar una función logística, *softmax*, o activación lineal dependiendo de la naturaleza de la aplicación [107].

El número de neuronas de la capa de salida corresponde al número de clases en las que se divide el conjunto de entrenamiento; y entonces el vector de salida se utiliza para encontrar la categoría a la que pertenece la imagen analizada. En términos más directos, las capas FC actúan como un clasificador dentro de las CNN. Los pesos de las capas FC, son actualizados usando retropropagación con descenso de gradiente. Cuando tenemos un modelo con muchos parámetros y una cantidad reducida de imágenes del conjunto de datos; generalmente su usa las técnicas de "Normalización" y *"dropout"*, cuyo propósito fundamental es evitar el sobreajuste del modelo [31], [113].

Una de las primeras redes neuronales convolucionales, basada en la distribución de pesos, se llamó *LeNet-5* [58]. Esta red fue utilizada en los bancos para identificar números escritos a mano en los cheques. Después de esa red, no hubo muchos cambios en la estructura, pero se incrementaron el número de capas y nuevas funciones de activación como ReLU.

El incremento en potencia computacional ayudó mucho a obtener mejores resultados, sobre todo cuando se entrenaban redes muy profundas y grandes volúmenes de datos de entrada. Un factor que ha jugado un rol importante en incrementar la prominencia de las redes neuronales convolucionales ha sido la

competencia anual ImageNet[5], también llamada *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC, por sus siglas en inglés). Uno de los primeros modelos en lograr el éxito en la competencia de ImageNet en 2012, fue la red AlexNet [52].

1.12 Clasificación con redes neuronales convolucionales.

Generalizando, la clasificación de imágenes (y también de imágenes remotas) describe la imagen completa mediante la extracción de características, ya sea de manera manual o mediante métodos de aprendizaje y después usa un clasificador para identificar la categoría del objeto. Por lo tanto, la extracción de las características de la imagen se vuelve un proceso de lo más importante. Antes de las redes de aprendizaje profundo se utilizaban métodos manuales o semimanuales, como los descritos anteriormente. Estos métodos tradicionales además de contener los pasos de extracción y codificación de características más el diseño del clasificador; incluían además, aprendizaje de características de bajo nivel, codificación de características, ajustes espaciales, diseño del clasificador y la fusión del modelo [31].

El surgimiento de las CNN ha logrado una serie de avances en el campo de la clasificación de imágenes. A diferencia de los métodos tradicionales, los métodos basados en CNN son un solo proceso de aprendizaje de punta a punta, la entrada al proceso es únicamente la imagen con pocos ajustes; los procesos de entrenamiento y predicción son llevados a cabo en automático por la red neuronal, la cual entrega el resultado final. Con las CNN se abandonan los cuellos de botella de los métodos tradicionales ocasionados por la extracción manual de características.

5 La página oficial de esta competencia se puede visitar en: https://www.image-net.org/challenges/LSVRC/.

1.13 Primeros modelos de CNN.

En 1998, Le Cun et al. [58] construyeron la red LeNet-5, cuyo enfoque era que se pueden construir mejores sistemas de reconocimiento de patrones cuando se utiliza el aprendizaje automático en vez de los modelos heurísticos de los diseños manuales. Usando el caso de estudio del reconocimiento de caracteres, se muestra que la extracción de características se puede llevar a cabo mediante el diseño cuidadoso de máquinas de aprendizaje que operan directamente sobre los píxeles de la imagen.

La CNN LeNet-5 comprende siete capas, sin contar la entrada. Las imágenes de entrada tienen dimensiones de 32 x 32 píxeles. La capa C1 es una capa convolucional de seis mapas de características de 28 x 28, que utilizan filtros de 5 x 5 conectados a la imagen de entrada. La capa S2 es una capa de reducción de dimensiones con seis mapas de características de tamaño 14 x 14. Esta etapa de muestreo utiliza filtros de 2 x 2 sin empalme, lo que ocasiona que se tengan la mitad de columnas y renglones en el mapa de características de esta etapa. El esquema anterior se repite para las capas C3 que tiene 16 mapas de características de tamaño 10 x 10 y la capa S4 que contiene los mismos 16 mapas de características pero de 5 x 5, que es la mitad del tamaño de la paca C3. Después la red continúa con la capa C5 que es otra capa convolucional con 120 mapas de tamaño 1 x 1, porque se utilizan filtros de tamaño 5 x 5 contra S4; de tal manera que S4 y C5 están completamente conectados. La capa F6 contiene 84 unidades y está completamente conectada a C5. La capa de salida contiene 10 unidades, donde una función de activación *softmax* calcula las probabilidades para cada categoría de la imagen de entrada [58].

Avanzando en el tiempo, para 2012, Krizhevsky et al. en [52] presentaron la CNN denominada AlexNet. Esta configuración fue la ganadora del premio ILSVRC de 2012 con una clara ventaja, es esta competencia se maneja un conjunto de

datos de entrada denominado ImageNet que contiene más de 15 millones de imágenes de tamaño considerable. Otra característica de esa red es que se utilizó un hardware mejorado con GPU cruzado, proporcionando un procesamiento en paralelo, ya que la red era demasiado grande para caber en la memoria de un solo GPU.

AlexNet contiene 8 capas y maneja cerca de 60 millones de parámetros. Es muy similar en concepto a LeNet, pero muy diferente en la configuración dada a cada capa. Contiene 5 capas convolucionales, 2 capas completamente conectadas (FC) y una capa de salida. Como las imágenes de entrada eran grandes en dimensiones, se volvía necesario el utilizar una primera capa convolucional de gran tamaño para manejar la extracción de características. Algunas de las mejoras que introdujo AlexNet son: la función de activación es cambiada de sigmoidea a ReLU; se usa una técnica de abandono (*dropout*) para reducir la complejidad del sistema; se usa también la técnica de aumento de datos para incrementar la cantidad de imágenes analizadas, a través de modificaciones de las imágenes existentes [31].

1.14 VGGNet

En 2014, Simonyan y Zisserman [53] propusieron el modelo de redes neuronales convolucionales denominado Grupo de Geometría Visual (VGG, por sus siglas en inglés) y lo utilizaron para ganar el concurso ILSVRC de ese mismo año. El modelo VGGNet, es similar al de AlexNet, en el sentido de usar un conjunto de capas convolucionales con reducción de dimensiones y luego una serie de capas completamente conectadas y terminar en una capa de salida que emplea *softmax* para generar las probabilidades de clasificación.

La contribución principal del trabajo de Simonyan y Zisserman es llevar a cabo una evaluación extensa de redes convolucionales que van incrementando su

profundidad (más capas). Las CNN habían ganado gran éxito en el reconocimiento a gran escala de imágenes y video, en especial utilizando el repositorio público de imágenes llamado ImageNet y la utilización de adelantos tecnológicos en el hardware de las computadoras como las unidades GPU. En la construcción de las redes VGG, se estudia la profundidad de la red al incrementar la cantidad de capas convolucionales, dicho incremento en la profundidad es factible debido al uso de filtros pequeños (3 x 3) en todas las capas.

La arquitectura de la red VGGNet maneja una capa de entrada de tamaño fijo, con imágenes RGB de 224 x 224. El único preprocesamiento de las imágenes es substraer el promedio del valor RGB de todo el conjunto de imágenes de entrenamiento, para cada píxel de la imagen en proceso. La imagen es pasada por varias etapas de capas convolucionales donde se usan filtros con campo receptivo muy pequeño (3 x 3), qué es la dimensión más pequeña para poder capturar la noción de izquierda, derecha, arriba, abajo y centro [53].

El “paso” (*stride*) de la convolución es fijado a un píxel; el “relleno” espacial (*padding*) es tal que la resolución espacial se mantiene después de la convolución. La “combinación” espacial (*pooling*) se lleva a cabo mediante cinco capas de combinación por máximos (*max-pooling*) [53].

Esta combinación se ejecuta con una ventana de 2x2 píxeles con un paso de 2 píxeles. Las etapas de capas convolucionales (que tienen diferente profundidad según su arquitectura) son seguidas por tres capas completamente conectadas (FC); las primeras dos tienen 4,096 canales cada una, la tercera capa FC tiene 1000 canales donde se lleva a cabo la clasificación de cada una de las clases, la última capa tiene la función *softmax*. La configuración de las capas FC es la misma para todas las configuraciones de la red VGG [53].

Las diferentes configuraciones de las redes VGG se muestran en la tabla 3; las cuales difieren solo en la profundidad. Desde 11 capas con pesos en la configuración A (8 capas convolucionales y 3 FC) y hasta 19 capas con pesos en la configuración E (16 capas convolucionales y 3 FC). El “ancho” de las capas convolucionales (el número de canales) es de alguna forma pequeño, empezando con 64 en la primera capa, y luego incrementando en un factor de 2 después de cada capa de combinación máxima, hasta alcanzar 512 canales. Las configuraciones VGG son diferentes a las usadas con éxito anteriormente; en vez de usar campos visuales relativamente grandes en las primeras capas convolucionales (p. ej. 11 x 11 en [52]) se usan campos receptivos de 3 x 3 en todas las redes. Se puede observar que el apilamiento de dos capas convolucionales de 3 x 3 nos proporciona un campo receptivo efectivo de 5 x 5 y si utilizamos tres capas convolutorias una detrás de otra, obtenemos un campo receptivo de 7 x 7.

El beneficio obtenido de unir varias capas convolutorias con un pequeño campo receptivo es que, primero, se incorporan 3 capas de rectificación no lineal en lugar de una, lo que hace a la función de decisión más discriminatoria. Segundo, se disminuye el número de parámetros; es decir, asumiendo que tanto la entrada como la salida de una pila de tres capas de 3 x 3 tiene C canales, toda esta etapa de capas convolucionales está parametrizada con $3(3^2C^2) = 27C^2$ pesos; al mismo tiempo, una sola capa convolucional con filtro de 7 x 7 requeriría $7^2C^2 = 49C^2$ parámetros, es decir 81% más [53].

El entrenamiento de la red se lleva a cabo optimizando la función objetivo con descenso de gradiente en lotes pequeños (basado en retropropagación [101]) con momento. El tamaño de los lotes se especifica en 256 y el momento en 0.9. Las muestras de entrenamiento se regularizaron con “degradación de pesos” (la penalización L_2 se definió en $5 \cdot 10^{-4}$) y regularización por “abandono” para las

primeras dos capas completamente conectadas (la proporción de abandono es de 0.5). La razón de aprendizaje se definió inicialmente en 10^{-2} y se fue decrementando en un factor de 10 cuando el conjunto de validación dejaba de mejorar. La inicialización de los pesos de la red es importante y la estrategia que se sigue en este tipo de redes es empezar con el entrenamiento de la red VGGNet de configuración A, que no es muy profunda y se pueden usar pesos aleatorios.

Configuraciones VGG					
A	**A-LRN**	**B**	**C**	**D**	**E**
11 capas con pesos	11 capas con pesos	13 capas con pesos	16 capas con pesos	16 capas con pesos	19 capas con pesos
Capa de entrada (imagen RGB de 224 $\times$ 224)					
Conv3-64	Conv3-64 LRN	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64
max-pooling					
Conv3-128	Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128
max-pooling					
Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv1-256	Conv3-256 Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv3-256 Conv3-256
max-pooling					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv1-512	Conv3-512 Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv3-512 Conv3-512
max-pooling					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv1-512	Conv3-512 Conv3-512 Conv3-512	Conv3-512 Conv3-512 Conv3-512 Conv3-512
max-pooling					
FC-4096					
FC-4096					
FC-1000					
softmax					

Tabla 3: Configuraciones VGG. La profundidad de la red se incrementa de izquierda (A) a la derecha (E), conforme se agregan más capas. La nomenclatura es “conv[tamaño del campo receptivo]-[número de canales]”. Fuente: [53].

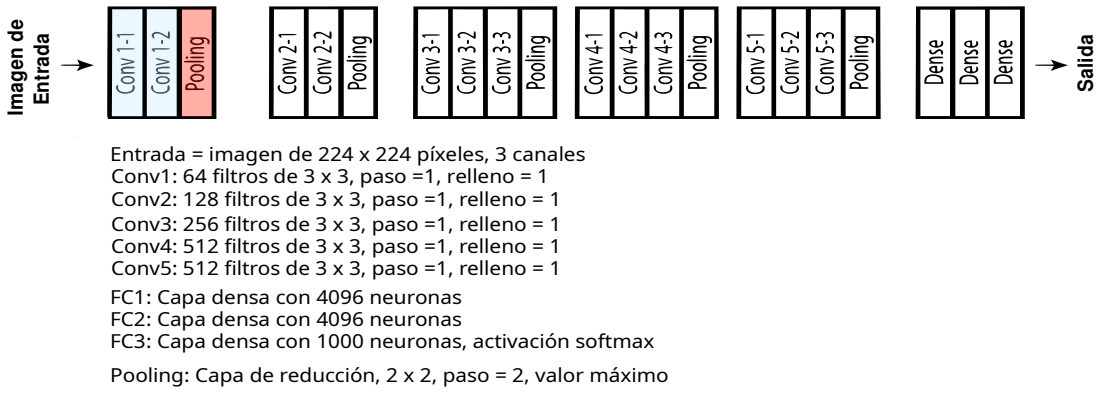


Figura 12: Versión simplificada de la red VGG-16, con 16 capas con pesos. Fuente: Elaboración propia.

Después, para entrenar redes más profundas, se inicializan las primeras cuatro capas convolucionales y las últimas tres capas FC con los pesos obtenidos del entrenamiento de la red de configuración A; las capas intermedias fueron inicializadas aleatoriamente. Los valores de sesgo (*bias*) fueron inicializados con cero. La imagen 12 muestra un esquema de la red neuronal VGGNet-16.

1.15 Red neuronal Inception.

En 2014, Szsegedy et al. [114] lograron la implementación de una red neuronal profunda y eficiente con el fin de utilizarla en trabajos de visión por computadora, que estaba basada en lo que ellos denominaron módulo "*Inception*" (que puede traducirse como origen, comienzo) y que a su vez este módulo fue derivado del concepto de "redes dentro de redes", presentado anteriormente en un artículo escrito por Li et al. en [115]. En este artículo se usan redes neuronales del tipo perceptrón multicapa (MLP, por sus siglas en inglés) para sustituir los filtros (*kernels*) dentro de las etapas convolucionales de la red neuronal y así lograr un mejor poder de abstracción del modelo.

El uso de estos módulos Inception incrementan la profundidad de la red neuronal. Primero porque se introduce una nueva forma de organización a través de los módulos Inception y, segundo, en el sentido más literal de hacer la red con mayor cantidad de capas.

1.15.1 Red Inception-V1.

La primera encarnación del concepto de módulo Inception fue la red neuronal GoogLeNet (Inception-V1). Donde la primera parte de la red es una red convolucional tradicional y a la que se le refiere como el “tallo”; la parte principal es la parte intermedia donde se colocan varios de los módulos Inception. La idea principal de estos módulos es que la información proveniente de las imágenes, esté disponible a través de diferentes niveles de detalle. Si usamos un filtro grande, podremos capturar información de un área más grande que contiene una variación limitada; si usamos un filtro más pequeño, podremos obtener información de un área más pequeña. De esta manera el modelo tiene la flexibilidad de capturar diferentes niveles de granularidad para la imagen. El módulo Inception realiza las convoluciones usando tres tamaños de filtros en paralelo. Los tamaños son 1 x 1, 3 x 3, y 5 x 5 [107].

La manera más directa de incrementar el desempeño de una red neuronal profunda es aumentar su profundidad, entendida como la cantidad de capas; y también su anchura, es decir, la cantidad de unidades o neuronas por capa. Sin embargo, esta solución directa trae dos problemas. Primero, se incrementa el número de parámetros, lo que conlleva el riesgo de sobre ajuste y segundo, se incrementa extremadamente la necesidad de recursos computacionales.

Otro de los propósitos de la arquitectura Inception, es reducir la dimensionalidad en las etapas donde se observe que los requerimientos computacionales se incrementan de tal forma que vuelven ineficiente al modelo. Entonces, se emplean convoluciones de 1 x 1 para realizar la reducción de

dimensiones antes de ejecutar las convoluciones de 3 x 3 y de 5 x 5 que requieren una mayor cantidad de procesamiento. El módulo Inception para GoogLeNet se muestra en la figura 13. En general, una red Inception, está integrada por módulos Inception del tipo mostrado en la figura 13 y que están apilados uno encima del otro, con algunas capas ocasionales de max-pooling con paso de 2 para reducir a la mitad la resolución de la malla [114].

En la red neuronal Inception-V1 (GoogLeNet) todas las convoluciones, incluyendo las que están dentro de los módulos Inception, usan activación ReLU. El tamaño del campo receptor de entrada es de 224 x 224 en el espacio de color RGB con media cero. En la tabla 4, la etiqueta “#3x3*” y “#5x5*” se refieren a la cantidad de filtros 1 x 1 en la capa de reducción antes de las capas convolucionales de 3 x 3 y de 5 x 5. Se puede conocer la cantidad de filtros 1 x 1 de la capa de proyección por la columna “pool proy.”.

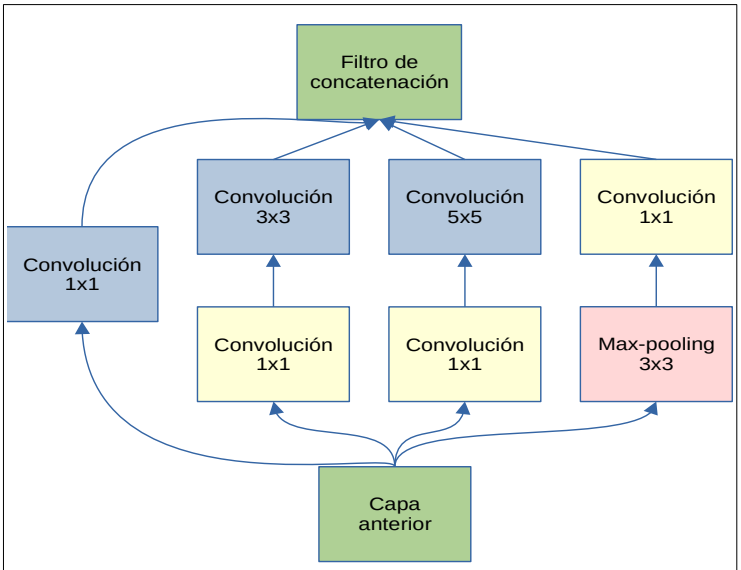


Figura 13: Módulo Inception-V1 con reducción de dimensionalidad. Fuente: [114].

Esta red se construyó con eficiencia computacional en mente, de tal forma que se puede ejecutar en computadoras aisladas, inclusive con poco poder computacional y memoria reducida. La red es de 22 capas de profundidad, si contamos capas con parámetros; 27 capas si contamos las de pooling. El número total de capas usadas para la construcción total de la red es de 100. La tabla 4 nos muestra la red neuronal profunda Inception-V1, donde se puede observar la utilización de la arquitectura Inception [114].

La configuración de esta red que se usó en la competencia ILSVRC 2014 se implementó con CPU únicamente, pero se pueden utilizar GPU, tal y como se llevó a cabo en la experimentación de este trabajo, pero cuidando la cantidad de memoria usada. El entrenamiento usa descenso de gradiente estocástico con un momento de 0.9. La razón de aprendizaje se va decrementando en 4% cada 8 épocas [48].

1.15.2 Red Inception-V3

La red neuronal Inception-V3 es un rediseño de la arquitectura Inception que proporciona eficiencia computacional y una cantidad menor de parámetros. Estas

Tipo	Filtro/paso	Tamaño	Capas	#1x1	#3x3*	#3x3	#5x5*	#5x5	Pool proy	Param	Ope
Convolución	7x7/2	112x112x64	1							2.7K	34M
Max pool	3x3/2	56x56x64	0								
Convolución	3x3/1	56x56x192	2		64	192				112K	360M
Max pool	3x3/2	28x28x192	0								
Inception (3a)		28x28x256	2	64	96	128	16	32	32	159K	128M
Inception (3b)		28x28x480	2	128	128	192	32	96	64	380K	304M
Max pool	3x3/2	14x14x480	0								
Inception (4a)		14x14x512	2	192	96	208	16	48	64	364K	73M
Inception (4b)		14x14x512	2	160	112	224	24	64	64	437K	88M
Inception (4c)		14x14x512	2	128	128	256	24	64	64	463K	100M
Inception (4d)		14x14x528	2	112	144	288	32	64	64	580K	119M
Inception (4e)		14x14x832	2	256	160	320	32	128	128	840K	170M
Max pool	3x3/2	7x7x832	0								

Inception (5a)		7x7x832	2	256	160	320	32	128	128	1072K	54M
Inception (5b)		7x7x1024	2	384	192	384	48	128	128	1388K	71M
Avg pool	7x7/1	1x1x1024	0								
Dropout (40%)		1x1x1024	0								
lineal		1x1x1000	1							1000K	1M
softmax		1x1x1000	0								

Tabla 4: Red neuronal Inception-V1 (GoogLeNet) donde se muestra la aplicación de la arquitectura Inception [114].

características permiten el uso de este modelo en las aplicaciones de visión móvil en presencia de escenarios big data. La gran reducción de parámetros de las redes Inception ha permitido el empleo de estas redes en proyectos que manejan el procesamiento de una gran cantidad de datos, aun en donde la capacidad de procesamiento y memoria están limitadas [32].

Muchas de las ganancias originales de la red Inception-V1 vienen del extenso uso de la reducción de dimensiones de la red, lo que a su vez permite mitigar el impacto en los cambios de la estructura de la red en Inception-V3. Esta red se usa como una red de visión por computadora, lo que hace necesario mantener la invariabilidad de la traslación y reemplazar los componentes completamente conectados por una arquitectura de dos capas convolucionales. La primera capa es una convolución de 3×3, la siguiente capa de 1×1 encima de la primera. Con este mecanismo se hace posible reemplazar una capa convolucional de 5×5, por dos capas convolucionales de 3×3. Otra implementación que se

maneja en la red Inception-V3 es la relacionada con la factorización de convoluciones a través del manejo de convoluciones asimétricas. Es decir, empleando una convolución de 3 $\times$ 1 seguida por otra convolución de 1 $\times$ 3 es equivalente a desplazar un filtro de dimensiones 3 $\times$ 3. El beneficio es que la solución de las dos capas asimétricas es de 33% más económica para el mismo números de filtros de salida [32].

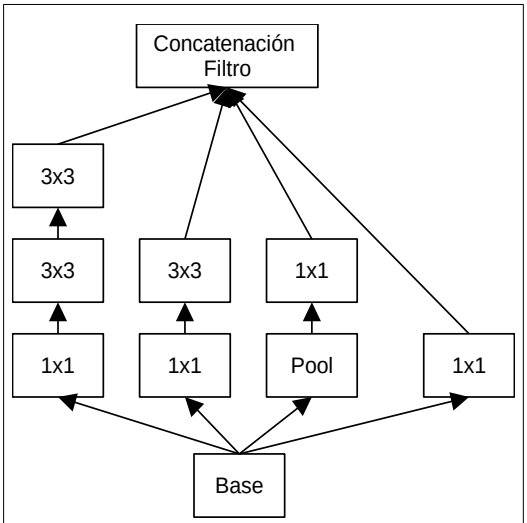


Figura 14: Módulo Inception-V3 donde se ha reemplazado un filtro 5x5 por dos 3x3. Fuente: [32].

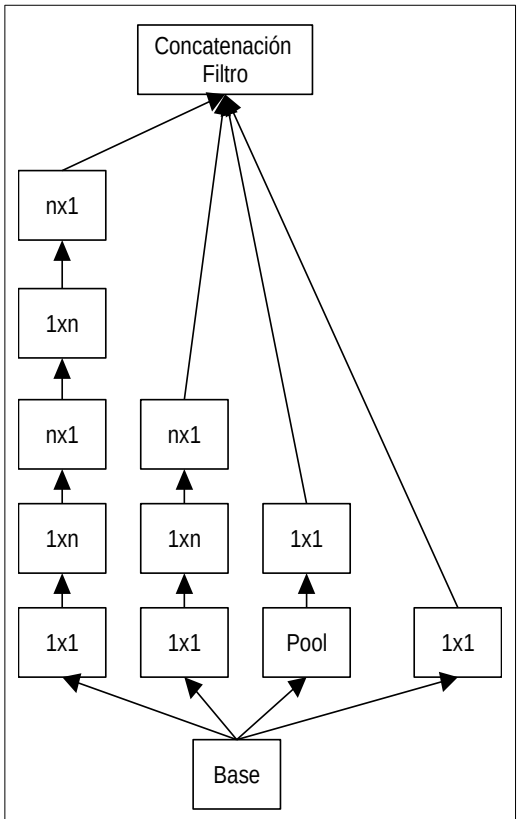


Figura 15: Módulo Inception-V3 después de la factorización de las convoluciones de 7x7 en la malla de 17x17. Fuente: [32].

La red Inception-V3 es una arquitectura mejorada. En las figuras 14 y 15 se muestran los dos tipos de arquitectura Inception utilizadas en esta versión del modelo; en la figura 14 se hace la sustitución de las convoluciones de 5 x 5 por

dos convoluciones de 3 x 3 y en la figura 15 se muestra la implementación de la factorización de convoluciones 7 x 7. El esquema de la red Inception-V3 se muestra en la tabla 5. Se observa que se ha factorizado la convolución tradicional de 7 x 7 en tres convoluciones de 3 x 3 y se aplican los conceptos de convoluciones factorizadas, asimétricas y más pequeñas que tienen como propósito el reducir el procesamiento computacional.

1.16 Red Neuronal Residual ResNet.

En 2015, se propuso la red neuronal profunda denominada ResNet por He et al. [116] y con ella determinaron un nuevo umbral en la clasificación de imágenes dentro de la competencia ILSVRC2015.

Como hemos visto en las redes neuronales anteriores, la idea general es avanzar en la cantidad de capas que se agregan a modelos más complejos, de tal suerte que se tiene la idea de que redes más profundas alcanzan mejor desempeño. Sin embargo, algunos experimentos han mostrado que el incrementar algunas capas en la red para hacerla más profunda, no pueden mejorar efectivamente el desempeño de la red neuronal [117].

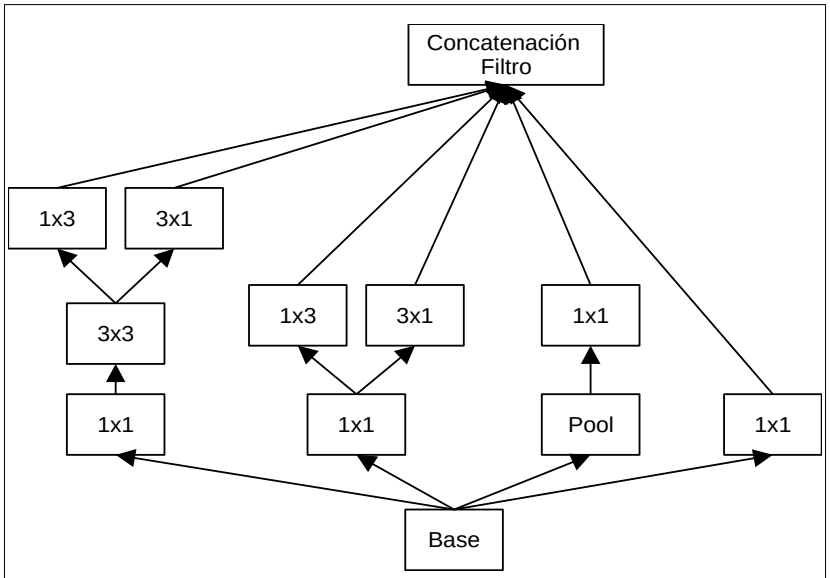


Figura 16: Módulo Inception-V3 con bancos de filtros expandidos. Se utiliza en las mallas de 8x8. Fuente: [32].

En [116] se menciona que el bajo desempeño al aumentar la profundidad de la red no se debe al efecto de desvanecimiento del gradiente o al sobre ajuste; ya que se han implementado redes con docenas de capas a las que se les añade una etapa de normalización, ya sea del tipo inicial o de lote, que fácilmente convergen usando retropropagación con descenso de gradiente estocástico. Entonces, se propone que la degradación de la red se debe a que en las redes muy profundas, no se logra una buena optimización, es decir, la optimización a través de SGD se vuelve difícil y se convierte en un serio problema. Las conexiones residuales en ResNet se usan para romper esa degradación y permitir que la red neuronal profunda alcance una alta exactitud.

El problema de la degradación se ataca con la propuesta de un marco de aprendizaje profundo residual. En vez de hacer que cada una de las pocas capas apiladas se ajusten directamente a un mapeo oculto predefinido, explícitamente se hace que estas capas se adapten a un mapeo residual definido. Formalmente, se

denota a la salida de la capa residual como $\mathcal{H}(x)$. Esta salida residual original se puede reescribir como $\mathcal{F}(x) + x$ [116].

Tipo	Filtro / Paso observaciones	Tamaño de entrada
Convolución	3x3/2	299x299x3
Convolución	3x3/1	149x149x32
Convolución con relleno	3x3/1	147x147x32
pooling	3x3/2	147x147x64
Convolución	3x3/1	73x73x64
Convolución	3x3/2	71x71x80
Convolución	3x3/1	35x35x192
3 $\times$ Inception	Como en la figura 14	35x35x288
5 $\times$ Inception	Como en la figura 15	17x17x768
2 $\times$ Inception	Como en la figura 16	8x8x1280
pooling	8x8	8x8x2048
Lineal	Logits	1x1x2048
Softmax	Clasificación	1x1x1000

Tabla 5: Estructura de la red Inception-V3, el tamaño de salida de cada módulo es el tamaño de entrada del siguiente. Se usa una convolución con relleno para mantener el tamaño de la malla. Fuente: [32].

La función $\mathcal{F}(x) + x$ se interpreta como “conexiones de corto circuito” que se implementan en algunas capas de las redes neuronales con alimentación hacia adelante, ver figura 17. Una conexión de corto circuito es aquella que salta una o más capas del modelo. En el caso de ResNet, la conexión de corto circuito realiza

una operación de “identidad” (la salida es igual a la entrada) y la salida de esta operación es sumada a la salida de las capas apiladas del flujo principal (figura 17). Las conexiones de identidad no agregan parámetros ni carga computacional adicional. La red completa puede ser entrenada de extremo a extremo con SGD y retropropagación [116].

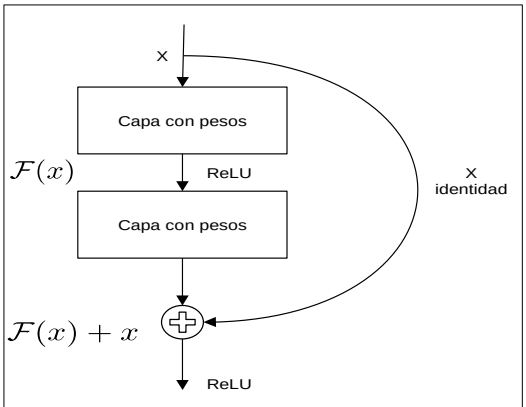


Figura 17: Módulo de aprendizaje residual, utilizado dentro de ResNet. Fuente: [116].

A través de diversos experimentos empleando la base de imágenes ImageNet, He et al. [116] concluyen que: 1) La red ResNet, en sus versiones más profundas (50, 101, 152 capas) son fáciles de optimizar a diferencia de aquellas de la misma profundidad, pero que carecen de conexiones de identidad. 2) Las redes ResNet pueden obtener fácilmente una ganancia en la medición de exactitud cuando se usan estas redes profundas.

La red residual está motivada por el problema de la degradación del gradiente y el aumento de la tasa de error. Entonces si las capas adicionales pueden ser construidas como mapeos “identidad”, se tiene que se podría entrenar un modelo más profundo cuyo error de entrenamiento no fuera mayor al de una red de la misma cantidad de capas pero sin conexiones de brinco. El problema de la degradación nos sugiere que los algoritmos de optimización podrían tener

dificultades al buscar aproximar los mapeos identidad al utilizar múltiples capas no lineales. A través del aprendizaje residual, y con estructuras de identidad óptimas, los optimizadores conducen a los pesos de las múltiples capas hacia el valor de cero para lograr los mapeos de identidad [116].

La arquitectura de la red está inspirada en las redes VGG. Las capas convolucionales tienen mayormente filtros de tamaño 3 x 3 y siguen dos reglas de diseño. 1) Para el mismo tamaño de un mapa de salida, las capas tienen la misma cantidad de filtros. 2) Si el tamaño del mapa de características es reducido a la mitad, entonces el número de filtros se duplica para preservar la complejidad de la capa. Para la reducción del mapa de características se utiliza una convolución con paso igual a 2. La red termina con una capa reductora usando promedio global y con una capa de salida de 1000 unidades con *softmax*. A cada dos capas se insertan las conexiones de corto circuito que se usan directamente cuando el tamaño de la entrada es igual al de la salida. Cuando las dimensiones son diferentes se usa una capa convolutoria adicional de 1 x 1 sobre el trayecto de corto circuito para hacer empatar las dimensiones de la entrada y la salida y llevar a cabo la adición [116].

En las corridas de implementación (con el conjunto de imágenes ImageNet) se configuró la imagen de entrada con un tamaño de 224 x 224 píxeles y también se substraía la media a cada píxel. Se utiliza normalización en lote (BN) después de cada convolución, se inicializan los pesos y se usa SGD con un mini lote de 256 imágenes. La razón de aprendizaje inicia en 0.1 y es dividida por 10 cuando el error no mejora y el modelo es entrenado hasta por $60x10^4$ iteraciones. Se usa una caída de los pesos de 0.0001 y un momento de 0.9; no se utiliza la técnica de abandono (*dropout*). A continuación se muestra la tabla 6, donde se definen las estructuras de diferentes implementaciones de la red ResNet, usando diferentes profundidades (34, 50 y 101 capas) [116].

1.17 Evaluación del desempeño de los clasificadores.

Los algoritmos de clasificación descritos con anterioridad ofrecen el camino para la predicción en la clasificación de imágenes. Esta predicción asigna una clase a cada una de las imágenes de la muestra, e invariablemente se obtienen imágenes clasificadas correctamente y otras colocadas en la clase incorrecta. En este trabajo, se utiliza la clasificación multiclase, es decir, tenemos más de dos clases a la que puede pertenecer la imagen bajo análisis. El algoritmo de clasificación asigna la imagen a la clase con el mayor valor de probabilidad.

Nombre de la capa	Tamaño de salida	34 capas	50 capas	101 capas
Conv1	112x112	7x7, 64 /2		
conv2_x	56x56	3x3, max pool /2		
		$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \end{bmatrix}$ x 3	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix}$ x 3	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix}$ x 3
conv3_x	28x28	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \end{bmatrix}$ x 4	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix}$ x 4	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix}$ x 4
conv4_x	14x14	$\begin{bmatrix} 3\times 3, 256 \\ 3\times 3, 256 \end{bmatrix}$ x 6	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix}$ x 6	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix}$ x 23
conv5_x	7x7	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \end{bmatrix}$ x 3	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix}$ x 3	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix}$ x 3
	1x1	Pool promedio, 1000-d FC, softmax		
FLOP		$3.6x10^9$	$3.8x10^9$	$7.6x10^9$

Tabla 6: Diferentes arquitecturas de la red ResNet. Los módulos residuales se muestran en corchetes con el número de veces que se repiten. [116].

A través del tiempo se han diseñado una serie de métricas para medir el desempeño o calidad del clasificador cuando ejecuta su trabajo de clasificación. La medición del error de clasificación no resulta ser muy objetiva, ya que es muy dependiente del algoritmo de clasificación, así como de la configuración de los parámetros que usa dicho algoritmo. Para la medición del desempeño de los clasificadores, se han empleado las métricas que se basan en la cantidad de imágenes que han sido clasificadas correcta e incorrectamente. Esto nos da una

línea base donde podremos comparar a cada clasificador de forma equitativa [118].

Las métricas usadas en el presente proyecto son la exactitud, la precisión, el recall y la puntuación F1 (*f1 score*). Estas métricas se basan en la contabilización de los aciertos y errores de clasificación que son representados de forma resumida en la llamada matriz de confusión.

1.17.1 Matriz de confusión.

La matriz de confusión es una tabla cruzada donde se representan el número de ocurrencias entre dos o más clases de un experimento. Comúnmente, en los renglones de la tabla se colocan las clases que representan a los valores verdaderos o reales y en las columnas se colocan las clases que identifican a las ocurrencias que han salido de la predicción [119]. Debido a la naturaleza de los experimentos podemos tener una matriz de confusión para experimentos multiclase y otra para experimentos binarios (de solo dos clases). La tabla 7 muestra la matriz de confusión binaria y la tabla 8 muestra la multiclase.

		Predicción	
	Clases	**Positivo**	**Negativo**
Real	**Positivo**	N_{TP}	N_{FN}
	Negativo	N_{FP}	N_{TN}

Tabla 7: Matriz de confusión binaria. Fuente: [119].

Las cuatro cantidades fundamentales de las matrices de confusión que podemos encontrar cuando se llevan a cabo predicciones con un clasificador donde todas las clases son conocidas, se describen como:

Verdaderos positivos (N_{TP}): Son aquellas ocurrencias positivas que el clasificador reconoce como positivas.

Verdaderos negativos (N_{TN}): Son aquellas ocurrencias negativas que el clasificador reconoce como negativas.

Falsos Negativos (N_{FN}): Son las ocurrencias positivas que el clasificador reconoce como negativas.

Falsos positivos (N_{FP}): Son las ocurrencias negativas, pero el clasificador las reconoce como positivas.

Podemos agregar que el número de clasificaciones correctas es la suma de los verdaderos positivos más los verdaderos negativos. También podemos agregar que el número de errores totales es la suma de los falsos positivos y los falsos negativos [118].

Para el caso de la matriz de confusión de múltiples clases, el cálculo de cada una de las cuatro cantidades fundamentales se lleva a cabo por clase, es decir, tendremos que calcular las cuatro cantidades por cada una de las clases de la matriz de confusión, como se observa en la tabla 8.

<table>
<tr><th></th><th colspan="6">Predicción</th></tr>
<tr><td rowspan="7">Real</td><td>Clases</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td></tr>
<tr><td>A</td><td>N_{TN}</td><td>N_{FP}</td><td colspan="3">N_{TN}</td></tr>
<tr><td>B</td><td>N_{FN}</td><td>N_{TP}</td><td>N_{FN}</td><td>N_{FN}</td><td>N_{FN}</td></tr>
<tr><td>C</td><td rowspan="3">N_{TN}</td><td>N_{FP}</td><td colspan="3" rowspan="3">N_{TN}</td></tr>
<tr><td>D</td><td>N_{FP}</td></tr>
<tr><td>E</td><td>N_{FP}</td></tr>
</table>

Tabla 8: Matriz de confusión multiclase. Se observa la distribución de las cuatro cantidades fundamentales para una sola clase (clase B). Esta distribución se repite para cada una de las clases involucradas. Fuente: [119].

1.17.2 Métrica de la exactitud de la clasificación.

La exactitud de la clasificación (Acc) [118] se define como la frecuencia de los eventos correctamente clasificados por el algoritmo en relación con la cantidad total de eventos. Utilizando las cantidades definidas en la matriz de confusión, se expresa mediante la ecuación 16:

$$Acc = \frac{N_{TP} + N_{TN}}{N_{FP} + N_{FN} + N_{TP} + N_{TN}} \tag{16}$$

La razón de error E, ver ecuación 17, es la cantidad de fallos en la clasificación de eventos entre el total de eventos del experimento, podemos expresarlo como $E = 1 - Acc$ y se calcula como:

$$E = \frac{N_{FP} + N_{FN}}{N_{FP} + N_{FN} + N_{TP} + N_{TN}} \tag{17}$$

Durante la ejecución de los experimentos del presente proyecto se lleva a cabo una clasificación multiclase; en este caso el cálculo de la exactitud es adecuado si consideramos las cantidades totales de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos de todo el experimento y no hacemos el cálculo separando por clases. La exactitud nos muestra una medición global de toda la muestra y en qué medida hace la predicción correcta considerando todo el conjunto de datos. Por lo anterior, la exactitud es más adecuada cuando nos interesamos por los elementos individuales de todo el conjunto de datos y no de la cantidad de elementos por clase [119].

1.17.3 Precisión y Recall (Sensibilidad).

En algunas aplicaciones, los eventos negativos superan a los positivos por un amplio margen (o viceversa); cuando esto pasa, la métrica de exactitud de clasificación nos ofrece una visión que no es realista del desempeño en la clasificación. Esa idea se aplica a los procesos multiclase donde la cantidad de elementos para cada una de ellas es desigual, es decir, las clases están

desbalanceadas y es una situación que ocurre con regularidad en las aplicaciones reales. Como por ejemplo, la cantidad de pacientes que sufren de algún padecimiento específico son relativamente escasos cuando los comparamos contra el total de la población. En las aplicaciones de este tipo, la exactitud de clasificación y la razón de error no expresan nada razonable sobre la utilidad práctica del clasificador. Se requiere de un criterio que se enfoque en una clase que, aunque tenga pocos elementos, es importante para el estudio [118].

La precisión (ecuación 18) se define como la cantidad de verdaderos positivos divididos entre todos los eventos que el clasificador etiquetó como positivos ($N_{TP} + N_{FP}$). El valor del indicador se calcula así:

$$Pr = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{18}$$

Expresándolo de otra manera, la precisión es la probabilidad de que el clasificador esté correcto cuando etiqueta un elemento como positivo.

La métrica de Recall (Sensibilidad) (ver Ec. 19), nos proporciona la probabilidad de que un evento marcado como positivo, sea correctamente reconocido como positivo por el clasificador. La fórmula para su cálculo es:

$$Re = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{19}$$

En algunas situaciones, la precisión es más importante que el recall. Tomemos como ejemplo un sistema de recomendación en un sitio de comercio electrónico. Al evaluar su desempeño, se desea lograr una alta precisión, es muy conveniente que el cliente esté satisfecho con la recomendación del sistema, porque así el cliente confíe en el sistema de recomendación. En esta situación el Recall no importa porque no afecta que el sistema identifique un porcentaje bajo

de todos los artículos recomendados que de todas maneras le interesan al cliente [118].

Por el contrario, en otras situaciones el Recall es más importante. Es el caso de los diagnósticos médicos. Un paciente que sufre una enfermedad y es correctamente diagnosticado, representa un verdadero positivo. Un paciente que sufre una enfermedad, pero no es diagnosticado correctamente es un falso negativo. Situación que los médicos desean evitar y, por lo tanto, se busca un valor de recall bajo. Entonces el número de falsos negativos N_{FN} debe ser muy reducido lo que a su vez nos conduce, mediante la aplicación de la fórmula a un valor de Recall alto [118].

1.17.4 F_1 Score.

El utilizar dos métricas para evaluar el desempeño, en algunas ocasiones nos complica el poder concluir sobre el desempeño de un clasificador. Resulta más sencillo el usar una sola métrica para medir el desempeño de algún proceso. En el caso de la precisión y el recall se ha implementado una fórmula que combina estas dos mediciones y se conoce como F_β, (ecuación 20) que es una generalización de la medición F_1:

$$F_\beta = \frac{(\beta^2 + 1) \times Pr \times Re}{\beta^2 \times Pr + Re} \tag{20}$$

El parámetro $\beta \in [0, \infty)$, permite dar un peso relativo a una de las dos medidas. Sí $\beta > 1$, entonces se da más peso al recall. Sí $\beta < 1$, entonces la precisión tiene un mayor peso. La medida F1 le otorga un peso igual a la precisión y al recall (ecuación 21), entonces haciendo $\beta = 1$, tenemos [118]:

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re} \tag{21}$$

La medida F_1 es la media armónica de la precisión y el recall, representa ambas medidas simétricamente en un solo valor. El mayor valor para la medida F_1 es 1, cuando tenemos una precisión perfecta y el menor valor posible es 0, que se obtiene cuando la precisión o el recall son cero [119].

Capítulo 2

Metodología

Capítulo 2. Metodología

2.1 Proceso general de los experimentos

La metodología normalmente utilizada para la clasificación de imágenes envuelve las siguientes tareas globales: selección de un método de clasificación, selección de las bases de datos a analizar, preprocesamiento de imágenes, extracción de características, entrenamiento del modelo de clasificación, posprocesamiento y evaluación del desempeño del modelo [22], [120].

Dentro de las varias opciones de modelos de clasificación, en este trabajo nos hemos centrado en la utilización de las denominadas redes neuronales artificiales (ANN, por sus siglas en inglés); por una parte, usamos redes neuronales convolutorias (CNN, por sus siglas en inglés) para obtener las características diferenciadoras de las imágenes remotas y por otra, utilizamos redes neuronales completamente conectadas (FCNN, por sus siglas en inglés) para la identificación de la clase correspondiente a cada una de las imágenes analizadas.

El enfoque del presente trabajo es completamente práctico, implementando tres algoritmos basados en redes neuronales convolutorias, las cuales se denominan VGGNet-16, ResNet-50 e InceptionV3; estos a su vez, emplean los conjuntos de datos “UCM Land Use”, “Sydney” y RSICD que ya fueron descritos con anterioridad.

Entonces tenemos un total de nueve experimentos, resultantes de la combinación de los tres modelos de redes neuronales convolutorias y de los tres conjuntos de datos de imágenes remotas. Cada uno de estos nueve experimentos se han codificado y ejecutado en dos sistemas de cómputo diferentes desde el punto de vista de su capacidad de procesamiento. El primer sistema es la

plataforma Google Colab donde se ejecutaron cada uno de los experimentos utilizando procesadores del tipo GPU, mientras que el segundo sistema es una computadora personal que utiliza un procesador tipo CPU. Esta doble configuración nos permite obtener los tiempos de ejecución en cada tipo de procesador y además las medidas de desempeño relacionadas con la calidad en la tarea de clasificación para cada uno de los experimentos. Las características de los sistemas de cómputo manejados se definen en las tablas 9 y 10 para GPU y CPU respectivamente.

El propósito general del experimento es medir el desempeño de la tarea de clasificación para cada modelo, así como los tiempos de ejecución durante dicha tarea. Para cada combinación de modelo de red neuronal y conjunto de datos, a través de varias métricas y comparación de los resultados para conocer si existen diferencias en el desempeño y tiempos de ejecución de cada combinación de modelo y conjunto de datos.

2.2 Google Colaboratory

Cada uno de los modelos con procesamiento GPU, fue codificado y ejecutado en la plataforma en línea Google Colab[6]. Esta plataforma permite ejecutar código en lenguaje Python de manera directa, sin ninguna configuración, que permite tener acceso a recursos de Hardware de alta capacidad como las unidades de procesamiento gráfico (GPU) y unidades de procesamiento de Tensores (TPU) y también nos permite compartir nuestro trabajo con algunos compañeros seleccionados a través de la herramienta Google Drive.

Básicamente, es un cuaderno de Jupyter Notebook que ha sido modificado por Google para ser ejecutado en los servidores remotos de alta capacidad de procesamiento. Además, nos facilita la utilización del código de TensorFlow que es

6 Google Colab, disponible en: https://colab.research.google.com/.

una plataforma de código abierto para desarrollar modelos de aprendizaje computacional[7].

Aunado a lo anterior, la plataforma Keras es una interfaz de programación de aplicación (API) de alto nivel para la codificación de redes neuronales, escrita en Python y que se ejecuta apoyándose en la plataforma TensorFlow. Fue diseñada para facilitar una experimentación rápida y permitir la elaboración de prototipos de una manera sencilla y en corto tiempo. Keras acepta varios componentes de redes neuronales tales como capas densas, convolutorias y recurrentes, así como capas de abandono (*dropout*) entre otras [121]. La versión inicial de Keras utilizada para la ejecución de los experimentos de clasificación es la 2.8, pero es una plataforma de mucho crecimiento, por lo tanto, las versiones siguen progresando rápidamente conforme pasa el tiempo.

Las características de Hardware para los procesadores tipo GPU dentro de Google Colab, se pueden obtener con los comandos `!nvidia-smi` y con `!lscpu` dentro del documento de Jupyter Notebook[8], las características de hardware del procesador tipo GPU se presentan en la tabla 9.

La versión gratuita de Google Colab maneja las siguientes especificaciones de hardware al momento de realizar los experimentos.

7 Plataforma TensorFlow disponible en: https://www.tensorflow.org/guide/basics.

8 Las especificaciones técnicas de la plataforma Google Colab se pueden consultar en: https://medium.com/analytics-vidhya/the-google-colab-system-specification-check-69d159597417.

GPU	
Id número	0
Nombre	Tesla T4
Persistencia	Apagada
Memoria	15.109 GB

Tabla 9: Características del procesador GPU de Google Colab. Fuente: [122].

Por otra parte, las especificaciones de la unidad de procesamiento tipo CPU[9] para la computadora personal utilizada durante la clasificación de imágenes remotas son:

CPU	
Arquitectura	X86_64
Modelo	Intel Core i5 – 11400
Cores	6
Threads	12
Frecuencia Base	2.60 GHz
Cache	12 MB
Memoria RAM	16 GB

Tabla 10: Características de la unidad de procesamiento tipo CPU, de la computadora utilizada en los experimentos. Fuente: Elaboración propia.

9 Las características de los procesadores Intel se pueden consultar en: https://ark.intel.com/content/www/us/en/ark/products/212270/intel-core-i511400-processor-12m-cache-up-to-4-40-ghz.html.

En la versión gratuita de Google Colab, el uso del GPU y CPU se pueden extender hasta por 12 horas de ejecución después de las cuales la ejecución se corta. De manera similar, permite hasta 90 minutos de tiempo libre de ejecución.

Entonces para los experimentos que requieren la unidad de procesamiento tipo GPU, se ha empleado la que se encuentra disponible para todos los usuarios en la plataforma Google Colab. Para la unidad de procesamiento tipo CPU, se ha manejado una computadora personal que se tenía disponible y que consideramos que cuenta con las características comunes de hardware que se pueden encontrar en muchas otras computadoras personales de empleo cotidiano en casas, empresas y escuelas.

2.3 Bases de datos de imágenes remotas.

Las fuentes de datos necesarias para este proyecto deben de ser construidas con imágenes remotas, es decir, fotografías tomadas a gran altura por aviones, drones o satélites. Las imágenes remotas son diferentes a las imágenes tomadas normalmente por las personas en situaciones comunes, ya que las últimas tienen una referencia donde podemos identificar en donde se encuentra el suelo y el cielo en la fotografía, o podemos identificar si un objeto en la foto está invertido o en la posición correcta.

En el caso de las imágenes remotas, son tomadas a gran altura, donde las mismas fotos pueden sufrir cambios de altura, acercamiento o definición en píxeles, por lo que los sistemas de reconocimiento de las características de las escenas de las imágenes deben de tomar en cuenta estas circunstancias que la hacen más compleja.

Comúnmente este conjunto de imágenes surgen de la segmentación de imágenes más grandes en tamaño (dimensiones de alto por ancho) tomadas

desde un satélite. Posteriormente, son agrupadas en clases o categorías de acuerdo a los objetos que aparecen en cada una de las imágenes seccionadas.

El término “base de datos” se refiere al conjunto total de imágenes que han sido seccionadas y agrupadas en clases según los objetos o texturas que aparecen en ellas. Cada una de las imágenes de la base de datos está identificada con el nombre de la clase a la que pertenece; esto facilita la distribución al momento de la separación de las imágenes en el sistema de cómputo.

A continuación se describen las características más sobresalientes de cada una de las bases de datos que se utilizan en el proyecto. Se han seleccionado estas por su libre disponibilidad a través de internet, así como porque presentan características que las diferencian entre ellas y ayudan a poner a prueba el nivel de desempeño del clasificador.

2.3.1 Conjunto de datos “UCM Land Use”

Este conjunto de datos fue presentado por [8] ante la necesidad de contar con un conjunto de imágenes remotas que además tuvieran textos descriptivos. Se le ha denominado conjunto de datos de uso de suelo de UCM (Universidad de California en Merced)[10], o de manera breve base de datos “UCM” (UCM por sus siglas en inglés). Fue extraída manualmente de imágenes más grandes de mapas de áreas urbanas de Estados Unidos propiedad del servicio de Geología de ese país.

Contiene 2,100 imágenes en total, que son divididas en 21 clases relacionadas con el uso del suelo. Cada clase tiene 100 imágenes y cada imagen tiene unas dimensiones de 256x256 píxeles. Los archivos digitales están codificados es formato de imágenes etiquetadas “*Tag Image File Format*” (TIFF, por sus siglas en inglés) las cuales no pierden información al momento de la

10 La base de datos UCM se puede descargar desde http://weegee.vision.ucmerced.edu/datasets/landuse.html

compresión, cada una contiene tres capas de intensidad, cada capa para los colores primarios rojo, verde y azul. La resolución de cada píxel es de 1 pie cuadrado. Las 21 categorías son: agricultura, aeroplanos, parques de béisbol, playas, edificios, chaparrales, área residencial densa, bosques, carreteras, campos de golf, bahías, intersecciones, área residencial media densidad, parques de casas rodantes, pasos a desnivel, estacionamientos, ríos, pistas de aterrizaje, área residencial de baja densidad, tanques de almacenamiento y canchas de tenis. Como se menciona en [8] estas clases de imágenes distintas se han seleccionado unas, por respecto a su homogeneidad, otras respecto a la textura, otras por su homogeneidad respecto al color y otras más por su no homogeneidad entre sí, lo que lo hace un conjunto de datos rico en características.

2.3.2 Conjunto de datos Sydney

El conjunto de imágenes remotas denominado Sydney, fue elaborado por [123] en el mismo proyecto donde se elaboró UCM. La base de datos de imágenes Sydney, contiene un total de 613 imágenes de alta resolución espectral agrupadas en 7 clases escénicas diferentes. Se desarrolló para un proyecto de etiquetado automático de imágenes y, por lo tanto, cada imagen contiene 5 textos descriptivos que explican la escena vista en la imagen.

En [9] se describen más a detalle las características de este conjunto de imágenes remotas. Fue obtenida de una imagen satelital de gran tamaño de la ciudad de Sydney, Australia a través de la plataforma Google Earth. La resolución espacial de esta imagen original es de 0.5 m y presenta unas dimensiones de 18,000 píxeles de ancho por 14,000 píxeles de alto. Las siete clases en las que se separan las 613 fotografías remotas son: residencial, aeropuerto, prado, río, océano, industrial y pistas.

La gran imagen original fue dividida en imágenes más pequeñas de 500x500 píxeles, dando un total de 1,008 imágenes individuales que resultaron de

la división inicial. Pero solo a 613 de ellas, se les asignó un texto descriptivo, por lo tanto, esta cantidad de imágenes es la que oficialmente se incluye en el conjunto de imágenes Sydney.

2.3.3 Base de datos RSICD

El conjunto de datos sobre imágenes remotas y textos descriptivos (RSICD, por sus siglas en inglés), es la mayor base de datos de imágenes usadas en este proyecto y fue empleada en [10] para un proyecto de creación automática de textos descriptivos para imágenes remotas.

Fue construido empleando imágenes principalmente de Google Earth, como se describe en [82], pero también se aprovecharon imágenes de BaiduMap, MapABC, Tianditu. Las dimensiones de las imágenes se mantuvieron a 224x224 píxeles con varias resoluciones. El número total de imágenes en el conjunto de datos es de 10,921. Una característica importante es que la base de datos se construyó con una alta diversidad intraclase y una baja disimilitud entre clases. El conjunto de imágenes remotas, está disponible públicamente y se puede descargar desde GitHub[11] y Google Drive[12] [10].

RSICD se creó con 30 categorías de escenas aéreas, las cuales son: aeropuerto, baldío, campo de béisbol, playa, puente, centro comercial, iglesia, comercio, residencial denso, desierto, cultivo, bosque, industrial, prado, residencial medio, montaña, parque, estacionamiento, área de juego, estanque, puerto, estación de ferrocarril, complejo turístico, río, escuela, residencial escaso, plaza, estadio, tanques de almacenamiento y viaducto. La cantidad de imágenes aéreas en cada categoría varían mucho, van desde 220 hasta 420 imágenes por clase; es decir, las clases en este conjunto de datos están desbalanceadas. Véase tabla 2.

11 El conjunto de datos RSICD se puede descargar de la plataforma GitHub desde: https://github.-com/201528014227051/RSICD optimal.

12 El conjunto de datos RSICD, también se puede descargar desde: https://drive.google.com/open?id=0B1jt7lJDEXy3aE90cG9YSl9ScUk

Para visualizar algunas de las imágenes de cada conjunto de datos, se han extraído algunas muestras de cada una de las bases de datos y se han colocado en la figura 18.

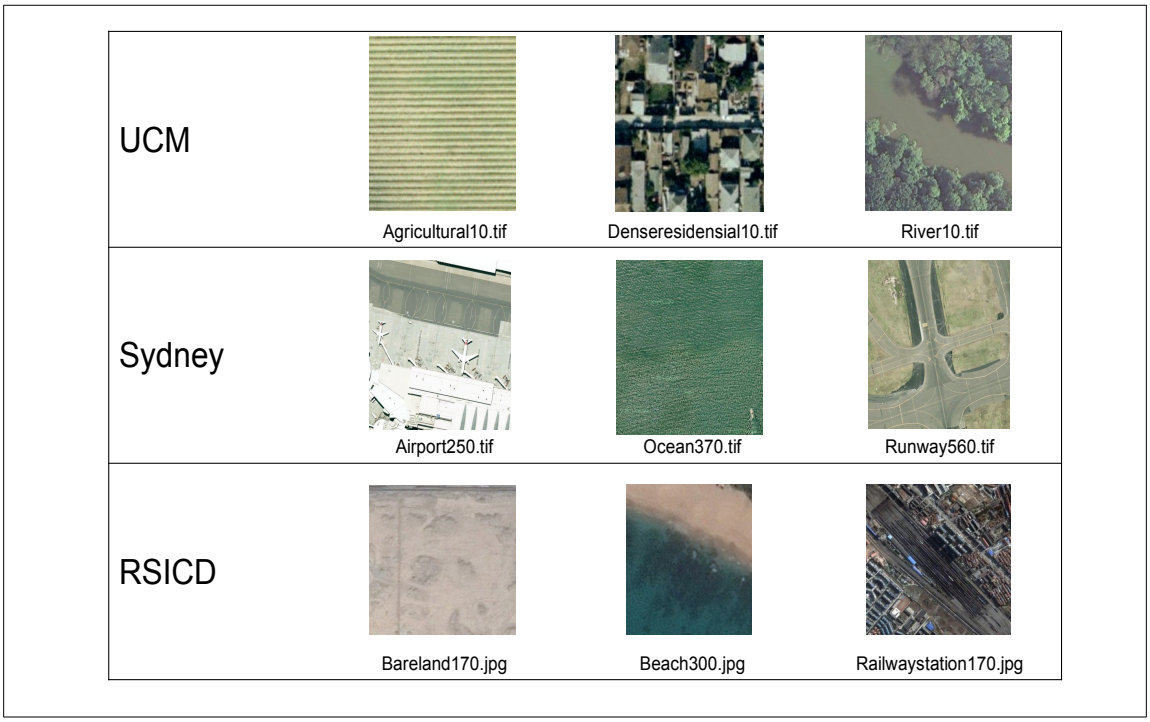


Figura 18: Ejemplos de imágenes remotas por conjunto de datos. Fuente: Elaboración propia.

2.4 Preprocesamiento de las bases de datos de imágenes

Antes de comenzar con el procesamiento del modelo para las imágenes que serán clasificadas, se debe de planear la distribución de las mismas en varias bolsas o subconjuntos con la finalidad de contar también con un número de imágenes que podrán ser utilizadas para la validación de la eficiencia del desempeño en los experimentos de clasificación.

En el aprendizaje de máquina uno de los principales requerimientos es la construcción de modelos con alto poder de predicción y generalización, lo cual se

puede lograr con una estrategia adecuada de segmentación de los datos. El problema de la segmentación del conjunto de datos a modelar se puede manejar a través del muestreo estadístico. El muestreo estadístico simple es un método muy común porque es eficiente y fácil de implementar. Las muestras son extraídas aleatoriamente de una distribución uniforme [124].

La totalidad de imágenes de cada uno de los tres conjuntos de datos que serán procesadas por los modelos de clasificación, se sometieron a una separación aleatoria en tres bolsas de imágenes, que se han denominado, entrenamiento, validación y prueba. La única diferencia que presentan estas bolsas de imágenes es en la cantidad de imágenes para cada una.

El conjunto de entrenamiento consta del 70% del total de imágenes, el conjunto de validación cuenta con el 20% y el conjunto de prueba con el 10% restante. En la tabla 11 se muestran las cantidades de imágenes por conjunto de datos y bolsa.

Base de datos	Entrenamiento (70%)	Validación (20%)	Prueba (10%)	Total
UCM	1470	420	210	2100
Sydney	429	123	61	613
RSICD	7645	2184	1092	10921

Tabla 11: Cantidad de imágenes por conjunto de datos y bolsa de modelado. Fuente: Elaboración propia.

La función del conjunto de entrenamiento es obtener un buen desempeño para el resultado de construcción del modelo de predicción, por esta razón cuenta con el mayor número de muestras. El conjunto de imágenes de validación se utiliza para verificar el poder de generalización sobre imágenes que no han sido vistas por el modelo, este conjunto de datos debe de contener las suficientes

imágenes para representar adecuadamente cada una de las clases del conjunto de imágenes. El grupo de imágenes de prueba lo dejamos reservado durante todo el proceso de entrenamiento y validación y lo aprovechamos hasta el final del experimento como un conjunto para la predicción y corroboración final del poder de predicción y generalización sobre imágenes nuevas.

La manera de implementar esta división del conjunto de imágenes es guardando por separado, cada uno de los tres conjuntos de imágenes en directorios diferentes de nuestro sistema de archivos. Entonces tenemos tres directorios plenamente identificados para el conjunto de entrenamiento, validación y prueba con los porcentajes de imágenes previamente descritos para cada base de datos.

El proceso de clasificación de imágenes empleando la plataforma de programación Keras necesita de una estructura específica para el manejo adecuado del conjunto de imágenes. Esta estructura se logra creando un directorio por cada clase en la que se divide el conjunto de datos. Entonces tenemos que dentro de cada uno de los directorios de entrenamiento, validación y prueba, se han creado una serie de subdirectorios que contienen una cantidad variable de imágenes, según el porcentaje previamente definido; cada uno de estos últimos directorios lleva por nombre el identificador de la clase a la que pertenecen las imágenes (ver Fig. 19).

Figura 19: Distribución del conjunto de datos en directorios de entrenamiento, validación y prueba. También se muestran los subdirectorios que representan las clases en las que se divide cada conjunto de imágenes. Fuente: Elaboración propia.

2.4.1 Generador de imágenes.

La plataforma Keras posee un mecanismo de manejo de imágenes que se encarga de la inyección de dichas imágenes al modelo de clasificación durante el entrenamiento. Keras selecciona aleatoriamente lotes de imágenes o genera

transformaciones de las imágenes existentes, que para el modelo están representadas como arreglos o matrices denominados tensores.

Estos lotes de datos de imágenes son procesados por el modelo de entrenamiento de redes neuronales convolutorias, uno tras otro hasta agotar la totalidad de imágenes del grupo de entrenamiento o validación. En cada procesamiento de cada lote se da un aprendizaje por la red neuronal.

Uno de los datos a suministrar durante la etapa de la generación de imágenes es el parámetro `bs`, que tiene que ver con la cantidad de imágenes por lote y se denomina "tamaño del lote" (*"batch size"*) y es conocido como un hiperparámetro en la construcción del modelo. Es un dato que es suministrado manualmente y para lograr la mejora del desempeño, se llevan a cabo varias ejecuciones del modelo cambiando el valor de la cantidad de imágenes por lote (`bs`), hasta encontrar un valor que maximice el desempeño del modelo, normalmente medido por la exactitud del conjunto de validación.

En esta etapa del proceso, las dimensiones (alto, ancho) de cada imagen pueden ser modificadas al momento de ser enviadas al procesamiento del modelo. El parámetro con el que se estandarizan las dimensiones de cada imagen se denomina tamaño de la imagen, `target_size`, y con él se especifican las nuevas dimensiones de alto y ancho de cada imagen.

Otro parámetro que ha sido suministrado manualmente al modelo es el denominado `rescale`, que se encarga de transformar cada valor de intensidad de las capas de la imagen en un valor de punto flotante entre 0 y 1.

El último parámetro que se ha definido en la etapa de generación de imágenes es el denominado `class_mode`, que se refiere al formato de la matriz de valores para cada etiqueta de clase. Este formato en las etiquetas de clase, es del tipo denominado *"one-hot"*, donde utilizamos el dígito 1 para identificar la clase

a la que pertenece la imagen y marcamos con 0 todas las demás clases. En todos los experimentos se definió como *"categorical"*, ya que necesitamos el nombre de cada clase por cada imagen procesada.

La tabla 12 muestra los hiperparámetros empleados para cada modelo de clasificación y conjunto de imágenes en los experimentos.

Parámetro	UCM	Sydney	RSICD
Tamaño de lote	32	16	32
Dimensión imagen	224x224	224x224	224x224
Re-escalado	1/255	1/255	1/255
Modo de las clases	categorical	categorical	categorical

Tabla 12: Hiperparámetros utilizados en la etapa de generación de imágenes para cada base de datos. Los parámetros fueron los mismos para cada uno de los modelos de clasificación, excepto para InceptionV3 con UCM donde usamos tamaño de lote igual a 16. Fuente: Elaboración propia.

2.4.2 Aumento de datos

Las bases de datos de imágenes remotas utilizadas no tienen características similares, las bases UCM DS y Sydney DS contienen una cantidad relativamente pequeña de imágenes para los estándares de clasificación con aprendizaje profundo. También Sydney DS y RSICD contienen una cantidad variable de imágenes en cada una de sus clases, algunas tendrán una cantidad adecuada de imágenes, pero otras clases contienen muy pocas imágenes, es decir, las clases están desbalanceadas.

Entonces aprovechamos la técnica de aumento de datos (*data augmentation*) para artificialmente incrementar la cantidad de imágenes de muestra para trabajar en los experimentos, además que nos permite mejorar el desempeño de los modelos cuando trabajamos con clases desbalanceadas [24].

Este incremento artificial de imágenes se lleva a cabo a través de la modificación o distorsión de las imágenes existentes. Los tipos de distorsiones que se aplicaron en los experimentos son del tipo de escala, transformando cada uno de los píxeles de la imagen a un valor entre 0 y 1 para cada una de las tres capas de intensidades RGB; de rotación, en donde se aplica un giro entre 0 y 360 grados a la imagen seleccionada; de acercamiento, donde se amplifica la imagen en un porcentaje específico y distorsión de giro sobre el eje vertical u horizontal.

Los parámetros para llevar a cabo el aumento de datos para cada uno de los experimentos fueron los mismos y están definidos en la tabla 13:

Parámetro	Valor
Rango de rotación	20
Rango de acercamiento	0.15
Giro vertical	True

Tabla 13: Parámetros utilizados en el aumento de datos. Fuente: [125].

Estos parámetros son definidos en el mismo comando que es utilizado para la definición de la generación de imágenes. Antes de que el lote de imágenes sea enviado al modelo de clasificación de aprendizaje profundo, las imágenes son transformadas para contar con más variaciones generadas de un conjunto de imágenes reducidas y aumentar la generalización del modelo.

2.4.3 Pre-entrenamiento del modelo

Cada uno de los nueve experimentos que surgen de la combinación de las tres redes neuronales convolutorias mencionadas anteriormente, utilizan un modelo previamente entrenado en un conjunto de datos externo y público denominado ImageNet.

El entrenamiento de un modelo de clasificación desde cero, inicia con pesos seleccionados aleatoriamente, es decir, pesos que no tienen ninguna relación con el conjunto de imágenes analizadas; el contar con un modelo preentrenado, ofrece la posibilidad de aprovechar los pesos no aleatorios, que ya fueron entrenados previamente en un conjunto de imágenes similares a las de nuestro problema.

El preentrenamiento de modelos se relaciona con la transferencia de aprendizaje donde lo ya aprendido con anterioridad nos sirve para llevar a cabo nuevas tareas donde incrementamos lo aprendido anteriormente. Este preentrenamiento es especialmente útil cuando trabajamos con redes neuronales profundas por su necesidad de una mayor cantidad de imágenes como insumo a nuestros modelos. Por otra parte, también el etiquetado de cada imagen lleva una gran cantidad de tiempo y posiblemente dinero, ya que para la tarea de clasificación de imágenes es necesario que cada una de nuestras imágenes estén previamente asignadas a una de las clases establecidas mediante una etiqueta y esto se lleva a cabo manualmente.

Para el preentrenamiento es necesario contar con un conjunto de imágenes que sean analizadas por algún modelo de redes neuronales convolutorias para así obtener un conjunto de pesos que puedan ser reutilizados en otra tarea de clasificación de imágenes.

El conjunto de imágenes más utilizado para el preentrenamiento es ImageNet[13]. Esta base de datos de imágenes está guiada por WordNet[14], que es a su vez, una base de datos de palabras en inglés. Entonces el objetivo general de ImageNet es tener una gran cantidad de imágenes por cada uno de los sustantivos que aparecen en WordNet, miles de imágenes por cada sustantivo en

13 El conjunto de imágenes ImageNet se puede consultar en: https://image-net.org/index.

14 La base de datos de palabras en inglés denominada WordNet, se puede descargar de: https://wordnet.princeton.edu/.

idioma inglés. Al momento de escribir este documento ImageNet cuenta con 14,197,122 imágenes en 21,841 palabras o clases.

La plataforma Keras ha facilitado las cosas para nosotros, ya que cada uno de los modelos a emplear en los experimentos con VGG-16, ResNet-50 e Inception-V3 ya han sido preentrenados con ImageNet. Esto es de gran ayuda porque incrementa el poder de predicción de los modelos y reduce el tiempo de entrenamiento.

En cada uno de los modelos en los nueve experimentos se han codificado los parámetros necesarios para llevar a cabo el pre-entrenamiento con ImageNet.

2.5 Redes neuronales utilizadas en los experimentos

Se seleccionaron tres redes neuronales convolutorias como modelos de clasificación para la ejecución de los experimentos. Como uno de los objetivos generales del experimento es verificar las diferencias existentes entre dichos modelos de redes neuronales, se escogieron tres de ellas por sus características que las hacen diferentes.

Se decidió usar solo tres modelos de redes convolutorias porque con esa cantidad, se muestran suficientes diferencias y también el agregar más modelos, incrementa la cantidad de experimentos totales y, por lo tanto, el tiempo empleado para su ejecución.

El primer modelo es VGGNet-16 elaborado por el grupo de geometría visual de la Universidad de Oxford Inglaterra (VGG, por sus siglas en inglés), que es uno de los primeros modelos de redes neuronales convolutorias profundas, ya que está compuesto de 16 capas de pesos entrenables, y aparece definida como configuración D en el documento referenciado por [53].

El segundo modelo es la red neuronal convolutoria profunda ResNet-50 (*Residual Network*), que cuenta con 50 capas de pesos entrenables y su característica principal es que utiliza capas de “cortocircuito” para reducir la complejidad de procesamiento a la vez que se incrementa la cantidad de capas del modelo.

La tercera red neuronal implementada en los experimentos, es la denominada InceptionV3, que también es una red neuronal convolutoria con 48 capas (secciones) de profundidad, desarrollada por Google y que usa la factorización de convoluciones para reducir la complejidad de las operaciones dentro de la red neuronal.

Cada una de estas redes neuronales implementadas en los experimentos, en general se configuraron sin alterar las capas convolutorias predefinidas en los diseños originales. En algunos de los modelos se modificó la cantidad de capas completamente conectadas o la cantidad de neuronas en cada capa, con la finalidad de mejorar la medida de exactitud del modelo.

2.6 Configuración de las capas de los modelos.

Los tres modelos de clasificación utilizados en los experimentos contienen un conjunto de capas convolutorias, seguidas por otra serie de capas completamente conectadas (*fully connected*). En la sección de capas completamente conectadas también encontramos otras capas que ayudan a reducir el sobre ajuste, como pueden ser capas de abandono (*dropout*) y capas de normalización por lotes (BN).

El entrenamiento de las capas convolutorias en nuestros ejercicios se llevan a cabo usando preentrenamiento y se hace uso de los pesos o mapas de características obtenidos como resultado de dicho proceso. Estos mapas de características resultantes son entregados a las capas completamente conectadas

del modelo para obtener como resultado final la clase a la que pertenece la imagen.

A continuación, en la tabla 14, se muestran el tipo y cantidad de capas empleadas en cada uno de los modelos de clasificación de los experimentos.

Modelo	**Capas Convolutorias**	**Capas completamente conectadas**	**Capas de Normalización y combinación**
VGG-16	13	3	3
ResNet-50	48	2	3
Inception-V3	46	2	1

Tabla 14: Cantidad de capas por modelo de clasificación. Fuente: Elaboración propia.

La plataforma de programación Keras, ya cuenta con un conjunto de funciones que facilitan la codificación de la sección de preentrenamiento de los tres modelos utilizados. En este trabajo en particular, se han implementado las funciones `VGG16`, incluida dentro del paquete *`keras.applications.vgg16`*; la función *`ResNet50`*, dentro del paquete *`keras.applications.resnet`* y la función *`InceptionV3`* que se deriva del paquete *`keras.applications.inception_v3`*. Cada una de estas funciones corresponden a los modelos que llevan el mismo nombre.

Para implementar el preentrenamiento en cada uno de los experimentos de este trabajo, en la tabla 15 se han definido los parámetros que requieren por igual cada una de las tres funciones citadas en el párrafo anterior.

Parámetro	Valor
include_top	False
input_shape	(224, 224, 3)
weights	“imagenet”

Tabla 15: Parámetros y sus valores utilizados en la función de preentrenamiento en los tres modelos de clasificación. Fuente: [126].

Durante la ejecución de la función empleada en el preentrenamiento, se descargan desde un servidor de Google, todos los pesos entrenados con anterioridad para cada uno de los modelos por separado.

La función del parámetro `include_top = False` es la de eliminar las capas completamente conectadas del modelo de clasificación, dejando únicamente aquellas capas relacionadas con el proceso convolucional de la extracción de características de las imágenes.

En el parámetro `input_shape = (224, 224, 3)`, los primeros dos valores definen las dimensiones de cada imagen, alto y ancho. El tercer valor del parámetro, se refiere a la cantidad de capas que se manejan en cada imagen. Cada capa corresponde a uno de los colores primarios rojo, verde y azul (RGB por sus siglas en inglés). Las dimensiones de la imagen de 224 píxeles de alto y 224 píxeles de ancho son los que la función de preentrenamiento requiere, de tal forma que si las dimensiones de las imágenes son más grandes, estas tienen que reducirse como se ha realizado en la etapa de generación de imágenes.

Con el parámetro `weights = “imagenet”` se le instruye a la función de preentrenamiento que se carguen en memoria los pesos aprendidos previamente cuando se ejecutó el proceso de clasificación del conjunto de imágenes de ImageNet.

Para el caso de la red neuronal convolutoria (CNN por sus siglas en inglés) VGG-16 definida en [53], se utilizan dos capas completamente conectadas con 4096 neuronas cada una, y una función de activación del tipo “ReLU” (*Rectified Linear Unit*). Como capa de salida se emplea una capa completamente conectada con la cantidad de unidades (neuronas) que requiera el conjunto de datos que estamos procesando, por ejemplo a UCM le corresponden 21 neuronas de salida y al conjunto Sydney le corresponden 7 neuronas a la salida, que empatan con la cantidad total de clases del conjunto de imágenes. A la salida de las capas preentrenadas se ha colocado una capa de reducción de dimensiones del tipo *“GlobalMaxPooling”* e inmediatamente después de cada una de las dos capas completamente conectadas se ha insertado una capa de ajuste del tipo *“BatchNormalization”*.

Para la red neuronal CNN denominada ResNet-50, tenemos una mayor cantidad de capas convolutorias cuando la comparamos con la red VGG-16, pero solamente contiene 2 capas completamente conectadas, como se define en [116]. La primera capa completamente conectada contiene un total de 1024 neuronas y se basa en una función de activación del tipo “ReLU”. Como capa de salida tenemos una red completamente conectada con la cantidad de neuronas que requiera el conjunto de datos que estamos analizando y una función de activación del tipo *softmax*. A la salida del modelo preentrenado colocamos una capa de reducción de dimensiones del tipo *“GlobalAveragePooling”* y antes de la capa de reducción de dimensiones colocamos una capa de ajuste del tipo *“BatchNormalization”*, colocamos otra capa de ajuste después de la capa completamente conectada que tiene 1024 unidades.

Para el tercer modelo utilizado que se denomina Inception-V3, la configuración de las capas completamente conectadas se ha llevado a cabo según se especifica en [32]. Después de la definición de la capa de preentrenamiento del

modelo, le sigue una capa completamente conectada con 1024 neuronas que usa una función de activación del tipo “ReLU”. La capa de salida es también una red densa cuya cantidad de neuronas dependen de la cantidad de clases que maneje el conjunto de datos que estamos analizando; esta capa de salida emplea una función de activación del tipo *softmax*. A la salida del modelo preentrenado se ha colocado una función de reducción del tipo *“GlobalAveragePooling”* e inmediatamente después de la capa densa de 1024 neuronas, se ha colocado una capa de ajuste del tipo *“BatchNormalization”*.

En el Anexo 1, se presenta un resumen de las capas, su tipo y la cantidad de parámetros que cada una contiene, por cada uno de los nueve modelos empleados en los experimentos. Dependiendo de la cantidad de capas con pesos de entrenamiento, la cantidad de parámetros cambia, entonces tenemos una cantidad total de parámetros de entrenamiento diferente para cada combinación de modelo y conjunto de datos de imágenes usadas.

En la tabla 16, se muestra un resumen con la cantidad de parámetros entrenables que cada uno de los modelos manejados. Esta cantidad de parámetros entrenables nos da una idea de la complejidad del modelo y también de la cantidad de instrucciones de procesamiento que se ejecutan para cada uno de ellos.

Modelo	UCM DB		Sydney DB		RSIC DB	
	Totales	Entrena-bles	Totales	Entrena-bles	Totales	Entrena-bles
VGG-16	33,716,053	18,984,981	33,658,695	18,927,623	33,754,974	19,022,878
ResNet-50	25,719,701	2,125,845	25,705,351	2,111,495	25,737,118	2,139,166
Inception-V3	23,922,485	2,119,701	23,912,231	2,107,399	23,935,806	2,130,974

Tabla 16: Cantidad de parámetros totales y entrenables por experimento. Fuente: Elaboración propia.

En la Tabla 16 podemos observar que los modelos ResNet-50 e Inception-V3 tienen una cantidad de parámetros entrenables muy reducida, ya que la mayor cantidad de parámetros ya han sido preentrenados la capa destinada para dicho proceso. Para estos modelos, los parámetros entrenables son aproximadamente solo el 10% del total de parámetros. Para el modelo VGG-16 la cantidad de parámetros entrenables se reduce, pero no tanto como para los modelos mencionados anteriormente. En este modelo los parámetros entrenables son aproximadamente el 56% de los parámetros totales.

En este punto ya contamos con un modelo de clasificación de imágenes completo, que contiene una serie de capas convolutorias (*Conv2D*) y de reducción de dimensiones (*MaxPooling*) que se encargan de la extracción de características de las imágenes. Complementando el modelo, posteriormente tenemos unas pocas capas completamente conectadas que se encargan del procesamiento de los pesos del modelo y de la selección de la clase pronosticada.

2.6.1 Compilación de modelos

La siguiente etapa del proceso de clasificación de imágenes a través de redes neuronales profundas es la compilación del modelo para cada uno de los experimentos.

En la fase de compilación del modelo se definen los parámetros del tipo de optimizador que se usará, el tipo de pérdida que el optimizador tratará de reducir al máximo y también se definen las métricas de desempeño que deseamos que el modelo calcule durante la etapa de entrenamiento.

Al calcular el conjunto de pesos en cada época de la ejecución de la red neuronal, se intenta reducir el error de predicción. Este error se mide como una función de pérdida. Esta función puede tener más de un mínimo local y la función de optimización intenta encontrar el mínimo global de la función de pérdida.

La función de pérdida que utilizamos en los modelos, es entropía cruzada categórica (`categorical crossentropy`) que se utiliza cuando tenemos un problema de multiclases con la variable respuesta del tipo categórica.

El optimizador definido en la mayoría de los modelos es “descenso de gradiente estocástico” (*Stochastic Gradient Descent*, SGD, por sus siglas en inglés) a menos que se especifique uno diferente en otro modelo.

En todas las etapas del proceso de entrenamiento de los modelos de clasificación que se han descrito anteriormente, se requieren parámetros configurables que proporcionen flexibilidad a los modelos y también un camino para la mejora de los resultados esperados del entrenamiento.

Durante la etapa de entrenamiento se ha experimentado con una serie distinta de valores para algunos de los parámetros más importantes involucrados en el entrenamiento y preprocesamiento de las imágenes.

A continuación en las Tablas de la 17 a la 25 se muestran los distintos valores ejercitados para los parámetros seleccionados y cuál combinación de valores nos entregó la mejor medida de exactitud considerada como parámetro de desempeño. La mejor medida de exactitud se marca con letra en negritas.

Las medidas de desempeño del modelo para medir la calidad de las predicciones, así como las mediciones del tiempo de ejecución del modelo durante la etapa de ejecución de la red neuronal, fueron obtenidas utilizando los valores de los parámetros de la columna con mejor desempeño en la medida de exactitud del conjunto de validación.

Experimento 1: VGG-16 & UCM							
ID	**VU01**	**VU02**	**VU03**	**VU04**	**VU05**	**VU06**	**VU07**
Aumento de datos	Sí	Sí	Sí	Sí	No	No	Sí
Normalización de lote	Sí	No	No	No	No	No	Sí
Optimizador	SGD	SGD	SGD	SGD	SGD	RMSProp	SGD
LR[15]	0.01	0.01	0.001	0.1	0.1	0.1	0.01
BS[16]	32	32	32	32	32	32	32
Épocas	46	46	48	39	39	16	53
Exactitud[17]	0.8286	0.8333	0.7071	0.8524	0.8786	0.2381	**0.8929**

Tabla 17: Lista de hiperparámetros empleados en los experimentos modelando VGG-16 y conjunto de datos UCM. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

15 Tasa de aprendizaje (LR, por sus siglas en inglés)

16 Tamaño del lote de imágenes (BS, por sus siglas en inglés)

17 Se refiere a la exactitud calculada con el conjunto de imágenes de validación.

Experimento 2: VGG-16 & Sydney						
ID	**VS01**	**VS02**	**VS03**	**VS04**	**VS05**	**VS06**
Aumento de datos	Sí	No	No	Sí	Sí	Sí
Normalización de lote	No	No	No	Sí	Sí	Sí
Optimizador	SGD	SGD	SGD	SGD	SGD	SGD
LR	0.1	0.01	0.001	0.1	0.01	0.001
BS	16	16	16	32	16	16
Épocas	14	39	92	36	36	43
Exactitud	0.9091	0.9187	0.8943	0.9106	0.9187	**0.9350**

Tabla 18: Lista de hiperparámetros utilizados en el experimento con modelo VGG-16 y conjunto de datos Sydney. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

Experimento 3: VGG-16 & RSICD						
ID	**VR01**	**VR02**	**VR03**	**VR04**	**VR05**	**VR06**
Aumento de datos	No	No	No	No	No	Sí
Normalización de lote	No	No	Sí	Sí	Sí	Sí
Optimizador	SGD	SGD	SGD	SGD	SGD	SGD
LR	0.1	0.1	0.1	0.1	0.01	0.01
BS	32	64	32	32	32	32
Épocas	15	SD	61	32	34	56
Exactitud	0.1357	0.1214	0.8125	0.7870	0.8080	**0.8315**

Tabla 19: Lista de hiperparámetros empleados en los experimentos modelando VGG-16 y conjunto de datos RSICD. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

Experimento 4: ResNet-50 & UCM								
ID	**RU01**	**RU02**	**RU03**	**RU04**	**RU05**	**RU06**[18]	**RU07**[19]	**RU08**
Aumento de datos	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Normalización de lote	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Optimizador	Adam	Adam	Adam	Adam	Adam	Adam	Adam	SGD
LR	0.0001	0.0001	0.001	0.00001	0.0001	0.0001	0.0001	0.01
BS	32	16	16	16	16	16	16	16
Épocas	100	100	100	100	150	100	100	43
Exactitud	0.7286	0.7238	0.6595	0.6619	**0.7310**	0.6357	0.7119	0.6738

Tabla 20: Lista de hiperparámetros empleados en los experimentos modelando ResNet-50 y conjunto de datos UCM. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

18 Se remplazó una capa de GlobalAveragePooling2D por otra de GlobalMaxPooling2D

19 Se agregó una capa completamente conectada adicional con 512 neuronas y una capa adicional de BatchNormalization.

Experimento 5: ResNet & Sydney								
ID	**RS01**	**RS02**[20]	**RS03**	**RS04**	**RS05**	**RS06**	**RS07**	**RS08**[21]
Aumento de datos	No	No	No	No	No	Sí	No	No
Normalización de lote	No	No	No	No	No	No	No	No
Optimizador	SGD	SGD	SGD	SGD	SGD	SGD	SGD	SGD
LR	0.1	0.1	0.01	0.001	0.01	0.01	0.01	0.01
BS	32	32	32	32	16	8	8	16
Épocas	21	12	20	30	31	16	22	31
Exactitud	0.5289	0.3967	0.7438	0.5785	0.7934	0.7355	0.7851	**0.8099**

Tabla 21: Lista de hiperparámetros empleados en los experimentos modelando ResNet-50 y conjunto de datos Sydney. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

20 Se incrementó una capa tipo “dense” adicional al modelo (2 capas en total) y no hubo cambios.

21 Se deja una sola capa completamente conectada con 1024 neuronas.

Experimento 6: ResNet-50 & RSICD				
ID	**RR01**	**RR02**	**RR03**	**RR04**
Aumento de datos	No	No	No	Sí
Normalización de lote	No	No	No	Sí
Optimizador	SGD	SGD	SGD	SGD
LR	0.01	0.01	0.01	0.01
BS	28	32	64	32
Épocas	39	47	57	81
Exactitud	0.2420	0.2990	0.2850	**0.5555**

Tabla 22: Lista de hiperparámetros empleados en los experimentos modelando ResNet-50 y conjunto de datos RSICD. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

Experimento 7: Inception-V3 & UCM					
ID	**IU01**	**IU02**	**IU03**	**IU04**	**IU05**
Aumento de datos	No	Sí	Sí	Sí	Sí
Normalización de lote	No	No	No	No	No
Optimizador	SGD	SGD	SGD	SGD	SGD
LR	0.01	0.01	0.01	0.01	0.01
BS	32	32	16	16	16
Épocas	23	38	24	50	56
Exactitud	0.9000	0.9214	0.9286	**0.9381**	0.9310

Tabla 23: Lista de hiperparámetros empleados en los experimentos modelando InceptionV3 y conjunto de datos UCM. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

Experimento 8: Inception-V3 & Sydney		
ID	**IS01**	**IS02**
Aumento de datos	Sí	Sí
Normalización de lote	Sí	Sí
Optimizador	SGD	SGD
LR	0.01	0.001
BS	16	16
Épocas	65	50
Exactitud	0.9174	**0.9174**

Tabla 24: Lista de hiperparámetros empleados en los experimentos modelando InceptionV3 y conjunto de datos Sydney. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

Experimento 9: Inception-V3 & RSICD							
ID	**IR01**	**IR02**	**IR03**	**IR04**	**IR05**	**IR06**	**IR07**
Aumento de datos	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Normalización de lote	No	Sí	Sí	Sí	Sí	Sí	Sí
Optimizador	SGD	SGD	SGD	RMSprop	RMSprop	Adadelta	Adadelta
LR	0.01	0.01	0.001	0.001	0.0001	0.001	0.01
BS	64	32	32	32	32	32	32
Épocas	25	25	25	25	24	25	25
Exactitud	0.8465	0.8630	0.8240	0.8490	**0.8635**	0.5920	0.8400

Tabla 25: Lista de hiperparámetros empleados en los experimentos modelando InceptionV3 y conjunto de datos RSICD. Los mejores valores están marcados en negritas. Fuente: Elaboración propia.

2.6.2 Entrenamiento del modelo

Una vez definido el modelo de clasificación con redes neuronales y la segmentación del conjunto de datos en entrenamiento y validación; podemos entrar en la fase de entrenamiento del modelo. Esta fase se refiere a la ejecución del modelo de clasificación una y otra vez, usando lotes predefinidos de imágenes hasta completar todas las imágenes designadas para entrenamiento. A cada ejecución del conjunto completo de imágenes de entrenamiento se le denomina “época”. El modelo de clasificación se ejecuta por una cantidad predefinida de épocas.

Al final de cada época el modelo de clasificación calcula la exactitud pero sobre el conjunto de imágenes denominado de validación, que es diferente al de entrenamiento. La ejecución del modelo se detiene una vez que se alcanzan la cantidad de épocas predefinidas o cuando la exactitud sobre el conjunto de validación no mejora durante una serie de épocas consecutivas.

La plataforma Keras contiene lo necesario para ejecutar el entrenamiento de modelos de clasificación con redes neuronales artificiales. Como ya tenemos previamente definido el modelo, solo falta indicar que realice las iteraciones sobre el conjunto de entrenamiento previamente definido y también que utilice el conjunto de validación ya armado. Otro aspecto necesario para el entrenamiento, es indicar a la plataforma Keras la cantidad de épocas que deberá de ejecutarse el modelo. En los experimentos aquí descritos se ha especificado que las ejecuciones lleguen a un máximo de 100 épocas.

Adicionalmente, se puede especificar otras características durante el entrenamiento que ayuden en los análisis posteriores o que recorten el tiempo de entrenamiento. Es así como durante el entrenamiento del modelo hacemos uso de dos funciones de “retorno” (*callback*). La primera función, que se utiliza como punto de control; guarda el modelo y sus pesos en el disco duro del sistema de cómputo. Se registran cada vez que el conjunto de pesos es mejor que el anterior y para esta comparación empleamos la medida de exactitud del conjunto de validación. De tal manera que guardamos en disco nuestro mejor modelo para su evaluación posterior.

La segunda función de retorno que empleamos durante el entrenamiento tiene que ver con el paro anticipado de la ejecución del modelo. Si el valor de la exactitud de validación no sufre cambios por una cantidad predefinida de épocas, entonces la ejecución del modelo se detiene debido a que el modelo no mejora. En todos los experimentos hemos configurado esta función para que se detenga una vez que no se registra mejora durante el 20% del total de épocas del modelo, que para los experimentos realizados significa 20 épocas.

También nos interesa conocer la duración del tiempo de ejecución del entrenamiento para los diferentes experimentos, cuando los modelos se ejecutan en un procesador tipo GPU comparado con un procesador tipo CPU. Para esto

empleamos la función *time()* de Python que es una forma de medir el tiempo transcurrido entre dos eventos de un programa de computadora. Este comando mide la cantidad de segundos transcurridos entre dos eventos y entrega un número flotante. La medición de la duración del tiempo de ejecución comprende desde el inicio de la primera época hasta que el entrenamiento se detiene porque no hay mejora o porque se alcanzaron la totalidad de épocas.

La medición del tiempo de ejecución del proceso de entrenamiento es a escala de décimas de segundo, no necesita ser más detallada. Sin embargo, para eliminar el tiempo de preparación del GPU después de haber estado en estado de ahorro de energía y antes de iniciar la medición del tiempo de ejecución, se corre una sola época del modelo para que al terminar, podamos dar inicio a la medición del tiempo y al desarrollo de las 100 épocas de nuestro conjunto de entrenamiento.

2.7 Métricas de desempeño del modelo

El propósito general del proyecto es comparar el desempeño de diferentes modelos de clasificación de imágenes cuando son entrenados con conjuntos diferentes de imágenes remotas. Tenemos la posibilidad de observar el comportamiento de un modelo de clasificación cuando se hace trabajar con bases de datos de imágenes que poseen características diferentes. Y de manera análoga, tenemos un conjunto de imágenes que se emplea para entrenar a varios modelos de clasificación y observar el desempeño de los modelos bajo la operación de un mismo conjunto de imágenes.

Las métricas que se manejan en este proyecto para medir la calidad de nuestro experimento son las comúnmente utilizadas en los procesos de clasificación. Las métricas usadas en el presente documento son la exactitud, la precisión, el recall (sensibilidad) y el valor-F1 (f1-score).

En breve, entendemos a la exactitud de clasificación como la proporción de muestras correctamente clasificadas por el algoritmo en relación con el total de muestras. La métrica de precisión nos permite conocer la proporción de positivos verdaderos entre la cantidad total de muestras que el clasificador ha etiquetado como positivas. El recall (sensibilidad) nos permite conocer la proporción de muestras positivas verdaderas en proporción al total de muestras que son verdaderamente positivas. Y por último la métrica *f1-score*, como se definió con anterioridad, es el promedio armónico entre la precisión y el recall.

Aplicamos cada una de las métricas de medición a cada uno de los nueve experimentos. Cada experimento ha generado con anterioridad un modelo de clasificación con sus pesos únicos para cada mezcla de modelo y conjunto de datos.

Durante el cálculo de las métricas de desempeño de los nueve experimentos se ha utilizado el conjunto de imágenes de validación. Para una mayor confiabilidad de los resultados obtenidos, se ha empleado la técnica de validación cruzada estratificada (*stratified k-fold cross validation*); que divide el conjunto de imágenes en k secciones del mismo tamaño y se ejecuta el cálculo del indicador para cada uno de los lotes de datos y posteriormente se obtiene un promedio aritmético de los valores obtenidos por estrato para determinar el valor final del indicador. Cada grupo en el que se dividió el conjunto de entrenamiento conserva la proporción de imágenes por clase.

El proceso de la evaluación del desempeño de cada experimento comprende inicialmente el construir un archivo en donde incluimos una columna con la lista de todas las imágenes del conjunto de datos y en otra columna incluimos la clase a la que pertenece cada imagen remota. De esta manera el mecanismo de estratificación cruzada tiene a su alcance el universo de imágenes que se han designado para el proceso de evaluación. En específico se utiliza la

función *flow_from_dataframe* del paquete *ImageDataGenerator* dentro de Keras.

Posteriormente, este archivo con los nombres de imágenes y la clase de cada una, se pasa a la función de estratificación cruzada incluida en la plataforma Keras. Se especifica de inicio que el conjunto de validación se deberá de dividir en 5 secciones.

Después, declaramos mediante código, las cuatro métricas que se requieren para la medición del desempeño del modelo. Recordemos que tres de ellas, exactitud, precisión y recall, ya fueron definidas durante el proceso de compilación del modelo de clasificación. Y en esta etapa, mediante código se define la función ejecutada para la métrica de f1-score.

El modelo de clasificación previamente entrenado y que contiene tres de las cuatro métricas empleadas, se carga en memoria y se le adiciona la definición de la métrica f1-score. Entonces usamos la función *evaluate()* para obtener los resultados de la evaluación del modelo.

Con la información reunida hasta ahora, se implementa un ciclo que se ejecutará por cada una de las cinco secciones en las que se dividió aleatoriamente el conjunto de imágenes remotas. En cada ejecución del ciclo se generan un conjunto de entrenamiento y otro de validación con los cuales se calculan las mediciones de desempeño. De esta forma, tendremos cinco mediciones para cada métrica, que posteriormente, mediante otro paso adicional se reduce a una sola mediante un promedio aritmético.

Otra medición, que también forma parte de los experimentos, es la duración del tiempo de ejecución del entrenamiento del modelo de clasificación.

Especificamos el tiempo de ejecución como la duración en segundos que el programa tarda en procesar todas las épocas del modelo de clasificación hasta que se detiene la ejecución del entrenamiento, ya sea porque se han alcanzado la cantidad máxima permitida de épocas o porque se ha alcanzado una terminación temprana. Dentro del programa, se inicia la contabilización del tiempo, justo antes de la ejecución de la primera época (y después de la ejecución de una época para asegurarnos que las unidades de procesamiento GPU han salido del estado de ahorro de energía) y termina inmediatamente después de que el ajuste de modelo ha concluido. Debido a la manera en que está codificado el algoritmo, la duración de la ejecución del entrenamiento, también incluye el tiempo destinado a guardar en disco el modelo, cada vez que se registra una mejoría en los valores de la métrica de exactitud.

Cada experimento proporciona un tiempo de ejecución distinto, inclusive para cada corrida del mismo modelo y base de imágenes. Este tiempo de ejecución mencionado depende principalmente de los recursos de hardware involucrados en cada corrida del experimento. Cada uno de los nueve experimentos de la combinación de modelo y conjunto de datos, se han ejecutado tres veces y se presenta como resultado el promedio aritmético de estos tres valores.

Los resultados obtenidos de las métricas de desempeño y de los tiempos de ejecución se pueden observar en el capítulo de resultados.

2.8 Predicción de imágenes

El resultado de la etapa de entrenamiento es un modelo de clasificación de imágenes remotas, que nos ayude en el futuro a realizar predicciones de imágenes remotas nunca antes presentadas al modelo, es decir imágenes nuevas pero similares a las usadas en el entrenamiento.

Durante la segmentación del conjunto de imágenes remotas completo, se ha reservado una cantidad pequeña de imágenes, que en nuestro caso es del 10% del total, que le denominamos segmento de prueba y es el que se maneja en esta etapa de predicción, ya que son imágenes que no se han empleado durante ninguna etapa previa del proyecto.

La predicción de imágenes nuevas es un ejercicio para poner en práctica el modelo de clasificación construido y tener una idea real del poder de generalización del modelo encontrado.

Los pasos que se han seguido en la predicción de imágenes inician con la selección aleatoria de una imagen por clase. Es decir, para la base UCM tendremos 21 imágenes, una por cada directorio clase. Para la base de imágenes Sydney tendremos 7 imágenes en total y para la base RSICD tendremos un total de 30 imágenes aleatorias para predecir la clase a la que pertenecen. Y la idea es que sin conocimiento previo, el modelo nos clasifique correctamente la clase a la que pertenece la imagen seleccionada.

Después, continuamos con la carga en memoria del modelo previamente entrenado. Y enseguida preprocesamos las imágenes previamente seleccionadas aleatoriamente, es decir, ajustamos el tamaño de la imagen, convertimos los valores de intensidad de cada imagen a un rango entre 0 y 1, luego incrementamos las dimensiones del arreglo de acuerdo a como lo requiere el modelo de clasificación.

Posteriormente, llevamos a cabo la predicción de las imágenes remotas usando la función *`predict()`* de Keras. El resultado de dicha función es la clase a la que pertenece cada una de las imágenes que se quieren predecir. Por último se visualizan cada uno de los resultados de la predicción, donde se muestra la imagen, la clase real a la que pertenece y la clase que ha predicho el modelo.

Los resultados de la predicción de imágenes para cada uno de los nueve experimentos se muestran en el capítulo 3.

Capítulo 3

Resultados

Capítulo 3. Resultados

A continuación aparecen los resultados para la clasificación de imágenes remotas, obtenidos al ejecutar cada uno de los tres modelos de clasificación (VGGNet-16, ResNet-50 & Inception-V3) aplicados a su vez, a cada uno de los tres conjuntos de imágenes remotas (UCM, Sydney & RSICD). Dando como resultado nueve experimentos para los cuales se muestran el conjunto de valores de las medidas de desempeño y los tiempos de ejecución; para los modelos mencionados cuando los aplicamos al propósito de clasificación de imágenes. Recordemos que los modelos de clasificación han sido previamente entrenados en la base de datos ImageNet.

Las bases de datos a utilizar han sido adaptadas, como se ha mencionado en la sección de metodología, para la tarea de clasificación. Las imágenes de cada base de datos se han separado por clase y se han colocado dentro de un directorio distinto cada una de ellas; de tal forma que el directorio identifica la clase a la que pertenece cada fotografía. Las imágenes se han separado en 70% de ellas para el conjunto de entrenamiento, el 20% para el conjunto de validación y el 10% adicional para el conjunto de prueba final.

Las métricas de desempeño que se presentan a continuación se han ejecutado en la plataforma Google Colab, beneficiándose del procesador tipo GPU, con las características descritas en la sección de metodología. Los resultados nos otorgan una perspectiva de la calidad del proceso de clasificación que cada uno de los tres modelos realiza sobre cada uno de los tres conjuntos de imágenes remotas.

3.1 Resultados obtenidos para la clasificación de imágenes remotas

A continuación se presenta la tabla 26 con resultados obtenidos para la tarea de clasificación apoyándonos en la técnica de validación cruzada estratificada. La perspectiva principal es desde el punto de vista de cada uno de los modelos de clasificación. Para cada combinación de modelo y conjunto de datos, tenemos las métricas de desempeño de exactitud, precisión, recall y f1-score. Adicionalmente, se agrega una columna con la pérdida o error que nos entrega el modelo final. El rango de valores de desempeño del modelo que podemos obtener se expresan entre los valores de 0 y 1 (o entre 0 y 100% si lo expresamos en porcentaje); excepto los valores en la columna de *Pérdida* que pueden ser mayores a 1.

Modelo	Base de Datos	Pérdida	Exactitud	Precisión	Recall	f1_score
VGG16	UCM	0.0450	**0.9888**	**0.9913**	**0.9854**	**0.9892**
	Sydney	0.1545	0.9826	0.9826	0.9826	0.9865
	RSICD	0.0668	0.9785	0.9830	0.9747	0.9788
Resnet50	UCM	0.3691	0.8948	0.9363	0.8201	0.8751
	Sydney	0.1521	**0.9418**	**0.9498**	**0.9327**	**0.9542**
	RSICD	1.0293	0.7143	0.8169	0.6079	0.6909
InceptionV3	UCM	0.0789	**0.9818**	**0.9859**	**0.9722**	**0.9802**
	Sydney	0.1378	0.9596	0.9674	0.9395	0.9554
	RSICD	0.1301	0.9611	0.9730	0.9510	0.9615

Tabla 26: Resultados del proceso de clasificación de imágenes remotas agrupadas por modelo. En negritas aparecen los mejores resultados por modelo y métrica. Fuente: Elaboración propia.

De la Tabla 26 se puede observar que para el modelo de clasificación VGG-16, los mejores resultados de exactitud, precisión, recall y f1-score, se obtuvieron

con la base de datos de imágenes remotas de UCM. Sin embargo, de los125 resultados obtenidos del modelo VGG-16 con todos los conjuntos de datos, podemos mencionar que todos los valores de las métricas estuvieron por arriba del 97%, un valor muy alto para la clasificación de imágenes.

Pasando al modelo de clasificación ResNet-50, se presentan los resultados obtenidos de cada una de las métricas cuando se combina con cada una de las tres bases de datos utilizadas en los experimentos. Observamos que los mejores resultados para este modelo se obtienen cuando combinamos ResNet-50 con el conjunto de imágenes Sydney; donde obtenemos valores por arriba de 93% para cada una de las 4 métricas empleadas en la medición del desempeño del modelo. En específico obtenemos 95.4% en la medición de f1-score y 94.18% para la exactitud. En contraste, se menciona que los resultados obtenidos de este modelo con la base de imágenes RSICD, son los más bajos de todos los experimentos. Obteniendo 69.09% para la medición de f1-score y 71.4% en la medición de la exactitud del modelo.

Ahora, sobre el modelo Inception-V3, podemos observar que los mejores resultados de desempeño se obtienen cuando se aplica al conjunto de imágenes remotas UCM, logrando resultados por arriba del 97% para todas las cuatro métricas. El resultado para la métrica de f1-score fue de 98.0% y para la exactitud del modelo se obtuvo en promedio un valor de 98.2%. Este modelo hace buen trabajo de clasificación en los tres conjuntos de imágenes usados, la métrica con desempeño más bajo de todas con el modelo Inception-V3 fue la métrica de recall con un resultado de 93.95%, el cual es un resultado muy alto para la tarea de clasificación de imágenes.

La Tabla 27, nos muestra el promedio aritmético entre los tres conjuntos de imágenes por modelo de clasificación, y donde podemos observar que los mejores resultados se obtienen con el modelo de clasificación VGG-16 para cada una de las cuatro métricas de desempeño. En segundo lugar, se tiene al modelo de

clasificación Inception-V3, que muestra valores muy cercanos a VGG-16 y en tercer lugar, con una diferencia significativa, se encuentra ResNet-50, que presenta los desempeños más bajos de todos los experimentos. Mencionamos también que los resultados de las métricas de desempeño para los modelos VGG-16 e Inception-V3, son muy buenos, ya que obtenemos valores por arriba del 95% en la clasificación de imágenes remotas.

Modelo	Pérdida	Exactitud	Precisión	Recall	F1-score
VGG-16	0.0887	0.9833	0.9856	0.9809	0.9848
ResNet-50	0.5168	0.8503	0.9010	0.7869	0.8401
Inception-V3	0.1156	0.9675	0.9755	0.9542	0.9657

Tabla 27: Promedio de cada métrica de desempeño agrupado por modelo de clasificación. Fuente: Elaboración propia.

La Tabla 26 nos mostró los resultados de las métricas desde la perspectiva de los modelos de clasificación cuando los utilizamos para entrenar distintas bases de datos de imágenes remotas. La Tabla 28 nos permite visualizar los resultados desde la perspectiva de los conjuntos de imágenes remotas cuando se combina cada una con los modelos de clasificación.

Base de datos	Modelo	Pérdida	Exactitud	Precisión	Recall	f1_score
UCM	VGG16	0.0450	**0.9888**	**0.9913**	**0.9854**	**0.9892**
	ResNet-50	0.3691	0.8948	0.9363	0.8201	0.8751
	InceptionV3	0.0789	0.9818	0.9859	0.9722	0.9802
Sydney	VGG16	0.1545	**0.9826**	**0.9826**	**0.9826**	**0.9865**
	ResNet-50	0.1521	0.9418	0.9498	0.9327	0.9542
	InceptionV3	0.1378	0.9596	0.9674	0.9395	0.9554
RSICD	VGG16	0.0668	**0.9785**	**0.9830**	**0.9747**	**0.9788**
	ResNet-50	1.0293	0.7143	0.8169	0.6079	0.6909
	InceptionV3	0.1301	0.9611	0.9730	0.9510	0.9615

Tabla 28: Resultados del proceso de clasificación de imágenes remotas agrupadas por conjunto de imágenes remotas. En negritas aparecen los mejores resultados por conjunto de datos y métricas. Fuente: Elaboración propia.

La tabla 28 nos presenta los resultados de los experimentos bajo otro punto de vista. Ahora tomamos como referencia a los conjuntos de imágenes remotas y cuáles son los resultados de las métricas de desempeño cuando son procesadas por los diferentes modelos de clasificación.

Visualizando la información de la tabla 28 podemos ver que para los tres conjuntos de datos, los mejores valores para las cuatro métricas las obtiene el modelo de clasificación VGG-16; al parecer este modelo es el que mejor maneja los tres conjuntos de datos empleados en los experimentos, es decir, se adapta a las diferentes condiciones de cada uno de ellos.

El conjunto de datos UCM entrega los mejores resultados en exactitud, precisión, recall y f1-score cuando se entrena con el modelo VGG-16, comparado contra los otros modelos. Inception-V3, entrega mediciones de desempeño muy parecidas a VGG-16, pero en menor medida. El modelo ResNet-50 y el conjunto

de imágenes UCM, alcanzan un 89.48% en exactitud y un 87.51% en la métrica f1-score; muy por debajo de los valores alcanzados por VGG-16 que son de 98.88% en exactitud y 98.92% en f1-score.

Los resultados del conjunto de datos Sydney, alcanzan su máximo valor de desempeño cuando los modelamos con VGG-16, ya que logran en promedio un 98.26% en exactitud y un valor de 98.65% en la métrica de f1-score. El modelo Inception-V3 le sigue con un valor de 95.54% en la métrica de f1-score y en tercer lugar ResNet-50 con 95.42% en f1-score. Podemos observar que cuando utilizamos este conjunto de datos, las diferencias son pequeñas entre los valores de las métricas de desempeño para los tres modelos de clasificación.

El conjunto de datos RSICD es la base de imágenes más compleja de analizar, ya que cuenta con más de 10,000 fotografías remotas, distribuidas en 30 clases desbalanceadas con una diferencia muy marcada en la cantidad de imágenes por clase. Sin embargo, el modelo VGG-16 que consta de una red neuronal convolutoria de 16 capas, es el que de nuevo mejores resultados nos ofrece cuando se entrena una base de datos tan grande. El valor de la exactitud es de 97.85% y el f1-score alcanza el 97.88%. El modelo Inception-V3 le sigue de cerca con 96.11% y 96.15% en exactitud y f1-score respectivamente. La combinación de la base RSICD con ResNet-50 es la que peores resultados nos presentan de todos los nueve experimentos. Solo alcanza el 71.43% en la métrica de exactitud y un 69.09% en la métrica de f1-score.

Conjunto de datos	Pérdida	Exactitud	Precisión	Recall	F1-score
UCM	0.1643	0.9551	0.9712	0.9259	0.9482
Sydney	0.1481	0.9614	0.9666	0.9516	0.9653
RSICD	0.4087	0.8846	0.9243	0.8446	0.8771

Tabla 29: Promedio de cada métrica de desempeño agrupado por conjunto de imágenes remotas. Fuente: Elaboración propia.

En la Tabla 29 se presentan los valores para cada una de las métricas de desempeño, pero usando un promedio aritmético que agrupe los tres valores de los modelos de clasificación, de tal forma que tenemos un solo valor por conjunto de datos para cada métrica de desempeño. Entonces, podemos observar que los mejores resultados en las métricas de exactitud, recall y f1-score son para el conjunto de imágenes Sydney y para la métrica de precisión, el mejor valor lo ha obtenido el conjunto de imágenes UCM.

En términos generales, el conjunto de datos Sydney tiene mejor desempeño en tres de las cuatro métricas y en sentido opuesto, el conjunto de datos RSICD ha logrado los valores más bajos en las cuatro métricas de desempeño.

3.2 Comportamiento de las métricas de pérdida y exactitud durante el entrenamiento.

Durante la etapa de entrenamiento del modelo, el lote de imágenes designadas como conjunto de entrenamiento, se evalúan a través de la red neuronal profunda hasta terminar con todas las imágenes. Al procesamiento de todas las imágenes asignadas a entrenamiento y agrupadas en lotes, se le llama una época.

Este proceso aleatorio de entrenamiento del modelo llamado época, se repite una cantidad considerable de veces, tantas como sean necesarias para

lograr la estabilización de la métrica de pérdida del modelo, es decir hasta que ya no podamos obtener una mejoría en el poder de clasificación del modelo.

Al momento de la ejecución del entrenamiento del modelo de clasificación, se define una métrica para evaluar el modelo y decidir si existe mejoría en el desempeño. Normalmente, se utilizan la métrica de pérdida o error y la métrica de exactitud del conjunto de imágenes de evaluación.

En los nueve experimentos del presente trabajo, se empleó la métrica de exactitud de imágenes de validación, para medir el desempeño del modelo de clasificación para decidir si el modelo bajo análisis, ha logrado su máximo valor de exactitud.

A continuación podemos observar una serie de figuras donde se presenta el comportamiento de las métricas de exactitud y pérdida para cada uno de los nueve experimentos.

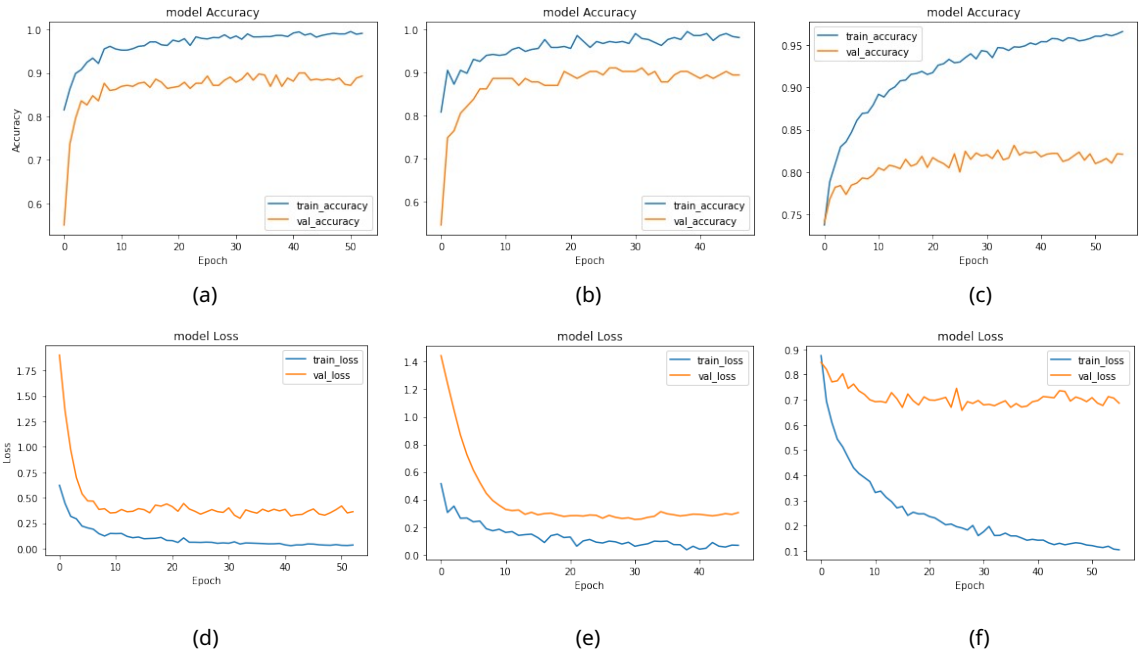


Figura 20: Desempeño del modelo de clasificación VGG-16 mostrando la exactitud y la p*érdida* conforme se ejecutan las *épocas en el entrenamiento*. Las gráficas (a), (b) y (c) muestran la métrica de exactitud para los conjuntos de imágenes UCM, Sydney y RSICD respectivamente. Las gráficas (d), (e) y (f) muestran la métrica de pérdida para los conjuntos de datos de UCM, Sydney y RSICD respectivamente. Fuente: Elaboración propia.

3.2.1 Comportamiento del modelo VGG-16

En la figura 20, se puede observar el comportamiento de las métricas de exactitud y pérdida, conforme se avanza en el procesamiento de las épocas durante el entrenamiento del modelo VGG-16.

La gráfica (a) nos muestra la exactitud para el conjunto de datos UCM, la gráfica (b) nos muestra la exactitud para el conjunto de datos Sydney y en la gráfica (c) observamos la exactitud para el conjunto de imágenes RSICD. La línea azul representa la exactitud calculada en el conjunto de datos de entrenamiento, la línea naranja representa la exactitud calculada en el conjunto de imágenes de validación.

Para la figura (a) vemos como la línea naranja, que representa la exactitud de validación, se estabiliza horizontalmente después de la época número 10, con

un valor cerca del 86%, vemos también que la exactitud de entrenamiento (línea azul) inicia en un valor cercano a 81% y continúa aumentando conforme se procesan las épocas hasta llegar a un valor de 97%. También la línea de la exactitud de entrenamiento tiende a estabilizarse porque el aumento de esta métrica es menor conforme avanzan las épocas; sin embargo, podemos observar que siempre continúa en crecimiento a lo largo de la ejecución de todas las épocas. Notamos también en la gráfica (a) que después de aproximadamente la época 25 las dos líneas se mantienen avanzando casi en paralelo, lo cual es un comportamiento deseable.

En la gráfica (d), se presenta la métrica de pérdida, también para el conjunto de imágenes de UCM. En esta gráfica observamos el comportamiento que sigue el conjunto de datos mencionado para los dos grupos de imágenes, el de entrenamiento en color azul y el de validación en color naranja. Idealmente, deseamos que la métrica de pérdida, llegue lo más cercano a cero. Se observa como para los datos de entrenamiento (línea azul), la línea desciende rápidamente hasta la época 20, donde después se observa que continúa decreciendo pero más lentamente. En lo que respecta a los datos de validación, la pérdida disminuye rápidamente hasta aproximadamente la época 10, donde se estabiliza y transcurre horizontalmente terminando en un valor de 0.4. Se observa también que después de la época 10, las dos líneas corren paralelas a través de todo el entrenamiento. La pérdida de validación termina en un valor de 0.31.

Continuando con la descripción de la figura 20, tenemos las gráficas (b) y (e) que representan la exactitud y la pérdida respectivamente, conforme transcurre la ejecución de las épocas de entrenamiento en el conjunto de datos Sydney.

La gráfica (b) nos presenta el comportamiento de la métrica de exactitud tanto para el conjunto de entrenamiento (línea azul), como para el conjunto de imágenes de validación (línea naranja). Podemos observar que aproximadamente después de la época 12, el conjunto de imágenes de validación se estabiliza

alrededor del valor 89%. También vemos que el conjunto de entrenamiento después de la época señalada, continua con un crecimiento lento hacia el sobre ajuste con gran variabilidad. Observamos como después de la época 12, la línea que corresponde al conjunto de validación, se mantiene avanzando en paralelo a la línea de entrenamiento, hasta que se obtiene un máximo valor para la exactitud de 91%.

La gráfica (e) nos muestra la métrica de pérdida para el conjunto de imágenes remotas Sydney. Esta gráfica tiene un comportamiento similar a la de exactitud, donde los valores de pérdida del conjunto de validación alcanzan un comportamiento horizontal después de la época 12, y se mantiene con ese comportamiento hasta que alcanzan un mínimo valor de pérdida de 0.3. Las gráficas de las métricas de exactitud y pérdida, para el conjunto de datos Sydney (gráficas b & e), presentan un comportamiento similar al conjunto de datos UCM (gráficas a & d).

El comportamiento de las métricas de exactitud y pérdida para el conjunto de datos RSICD, se visualizan en las gráficas (c) y (f) respectivamente. Este conjunto de datos es notoriamente más grande en cantidad de imágenes y también tiene la característica de contener la mayor cantidad de clases de imágenes, de los tres conjuntos de datos.

La gráfica (c) nos presenta el comportamiento de la métrica de exactitud conforme se ejecutan las épocas del experimento. Podemos observar que después de la época 15, el conjunto de datos de validación alcanza una estabilización horizontal, alcanzando un valor máximo de exactitud de 82%, la ejecución se detiene entre las épocas 50 y 60 porque ya no existe mejoría en esta métrica. También en esta misma gráfica podemos observar como la exactitud del conjunto de entrenamiento (línea azul) sigue aumentando, tendiendo hacia el sobre ajuste. De tal forma que la separación entre la línea de entrenamiento y la de validación, se hace cada vez más grande. Esto sugiere que todavía existe

espacio para una mejora de los valores de exactitud de validación, a través de la modificación de los parámetros de entrenamiento del modelo.

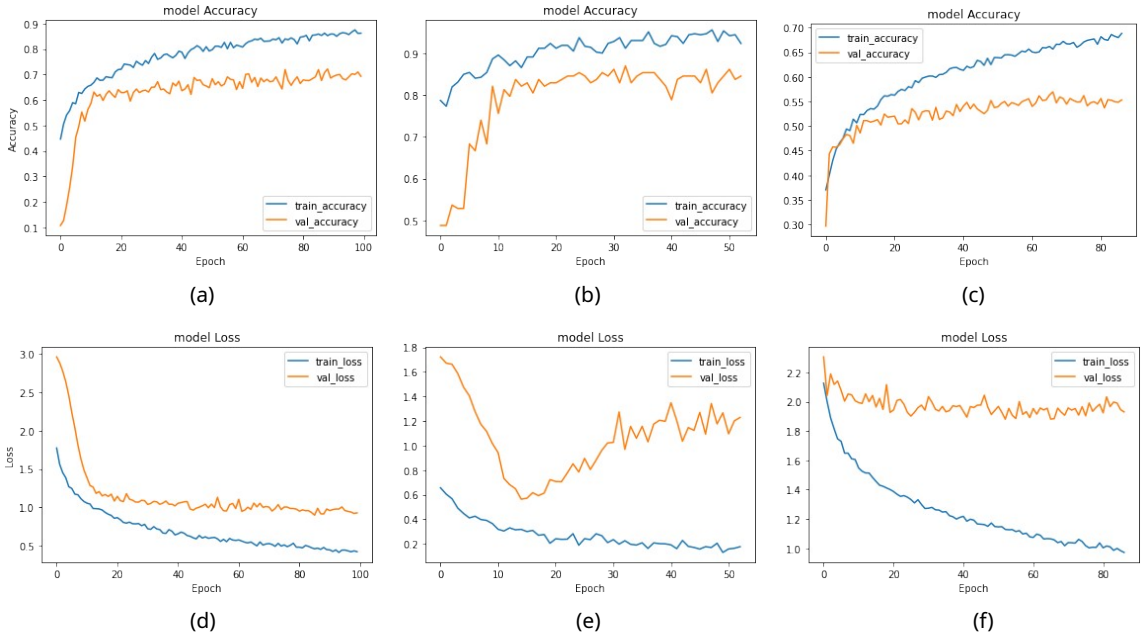


Figura 21: Desempeño del modelo de clasificación ResNet-50 mostrando la exactitud y p*érdida,* mientras se ejecutan las épocas en el entrenamiento. Las gráficas (a), (b) y (c) muestran la métrica de exactitud para los conjuntos de imágenes UCM, Sydney y RSICD respectivamente. Las gráficas (d), (e) y (f) muestran la métrica de pérdida para los conjuntos de datos de UCM, Sydney y RSICD respectivamente. Fuente: Elaboración propia.

3.2.2 Comportamiento del modelo ResNet-50

Ahora en la figura 21, se puede observar el comportamiento de los valores de las métricas de exactitud y pérdida, mientras se avanza en el procesamiento de las épocas durante el entrenamiento del modelo ResNet-50, para los tres conjuntos de datos.

Las gráficas (a), (b) y (c) nos muestran la métrica de exactitud para los conjuntos de datos UCM, Sydney y RSICD respectivamente. De manera análoga, las gráficas (d), (e) y (f) representan los valores de pérdida del modelo para los conjuntos de datos en el mismo orden. La línea azul representa la exactitud

calculada en el conjunto de datos de entrenamiento, la línea naranja representa la exactitud calculada en el conjunto de imágenes de validación.

Para la figura (a) vemos como la línea naranja, que representa la exactitud de validación, se estabiliza horizontalmente después de la época número 20, con un valor cerca del 61%, vemos también que la exactitud de entrenamiento (línea azul) presenta una pendiente positiva ligera, de crecimiento, a lo largo de todas las etapas de este experimento. Para la línea de la exactitud de entrenamiento vemos un ligero incremento a lo largo de las 100 épocas que ha durado este experimento; podemos observar que en este experimento de ResNet-50 con la base UCM, se han consumido el máximo de épocas de entrenamiento (100 épocas) y se observa que el crecimiento continúa a lo largo de la ejecución. Notamos también en la gráfica (a) que después de aproximadamente la época 20 las dos líneas se mantienen avanzando casi en paralelo.

Similarmente en la gráfica (d), se presenta la métrica de pérdida, también para el conjunto de imágenes de UCM. En esta gráfica observamos el comportamiento que sigue el conjunto de entrenamiento en color azul y el de validación en color naranja. Se observa como para los datos de entrenamiento (línea azul), la línea desciende rápidamente en el inicio de las épocas, y posteriormente se observa que continúa decreciendo más lentamente, hasta alcanzar el máximo de épocas de procesamiento. En lo que respecta a los datos de validación, la pérdida disminuye rápidamente hasta aproximadamente la época 20, donde continúa decreciendo lentamente y transcurre hasta no mostrar un mejor valor de 1.2. Se observa también que después de la época 20, las dos líneas corren con la misma dirección a través de todo el entrenamiento.

Continuando con la figura 21, observamos ahora las gráficas (b) y (e) que representan la exactitud y la pérdida respectivamente, conforme transcurre la ejecución de las épocas de entrenamiento en el conjunto de datos Sydney, que contiene 7 clases de imágenes.

En la gráfica (b) se presenta el comportamiento de la métrica de exactitud tanto para el conjunto de entrenamiento (línea azul), como para el conjunto de imágenes de validación (línea naranja). Podemos observar que aproximadamente después de la época 15, el conjunto de imágenes de validación, detiene su incremento y se estabiliza alrededor del valor 83%. También vemos que el conjunto de entrenamiento después de la época señalada, continua con un crecimiento lento hacia el sobre ajuste. Observamos como después de la época 15, la línea que corresponde al conjunto de validación, se mantiene avanzando en paralelo a la línea de entrenamiento, hasta que se obtiene un máximo valor para la exactitud de 83%.

La gráfica (e) nos muestra la métrica de pérdida para el conjunto de imágenes remotas Sydney. Esta gráfica tiene un comportamiento antagónico a la de exactitud, donde los valores de pérdida del conjunto de validación van en rápido decremento hasta la época 15, donde posteriormente empieza a incrementarse de nuevo hasta registrar valores cercanos a 1.3. Podemos observar en la gráfica (e) que después de la época 30 la variabilidad de los valores de la pérdida se incrementa haciendo que fluctúen considerablemente. Esta misma variabilidad la podemos ver en la gráfica (b) de la exactitud, donde después de la época 30 los valores de la exactitud fluctúan en mayor medida. La ejecución del entrenamiento del conjunto de datos Sydney termina anticipadamente después de la época 50.

El comportamiento de las métricas de exactitud y pérdida para el conjunto de datos RSICD, se visualizan en las gráficas (c) y (f) respectivamente. El comportamiento de estas dos gráficas es diferente a la de los conjuntos UCM y Sydney, posiblemente por lo diferente del tamaño del conjunto de datos y la diferencia en la cantidad de clases de RSICD.

La gráfica (c) nos muestra el comportamiento de la métrica de exactitud conforme se ejecutan las épocas del experimento. Podemos observar que la línea

azul del conjunto de entrenamiento continúa incrementando durante toda la ejecución de las épocas y que la línea de exactitud de validación, se estabiliza después de la época 20, alcanzando un valor máximo de exactitud de 52%. Además, podemos visualizar como las dos líneas, de entrenamiento y de validación, se van separando conforme avanzan las épocas del entrenamiento, esto sugiere que todavía existe espacio para una mejora de los valores de exactitud de validación. La ejecución se detiene entre las épocas 50 y 60 porque ya no existe mejoría en esta métrica.

Observando ahora la gráfica (f), nos percatamos que mientras la pérdida de entrenamiento continúa bajando constantemente a través de todas las épocas de entrenamiento, no es lo mismo para la pérdida de validación, la cual se estabiliza en valores alrededor de 2.0. Esto hace evidente la gran separación entre estas dos líneas del comportamiento de la pérdida, aludiendo que se podrían obtener valores de pérdida de validación aún menores mediante la modificación de los parámetros de entrenamiento del modelo o de las características del conjunto de datos RSICD.

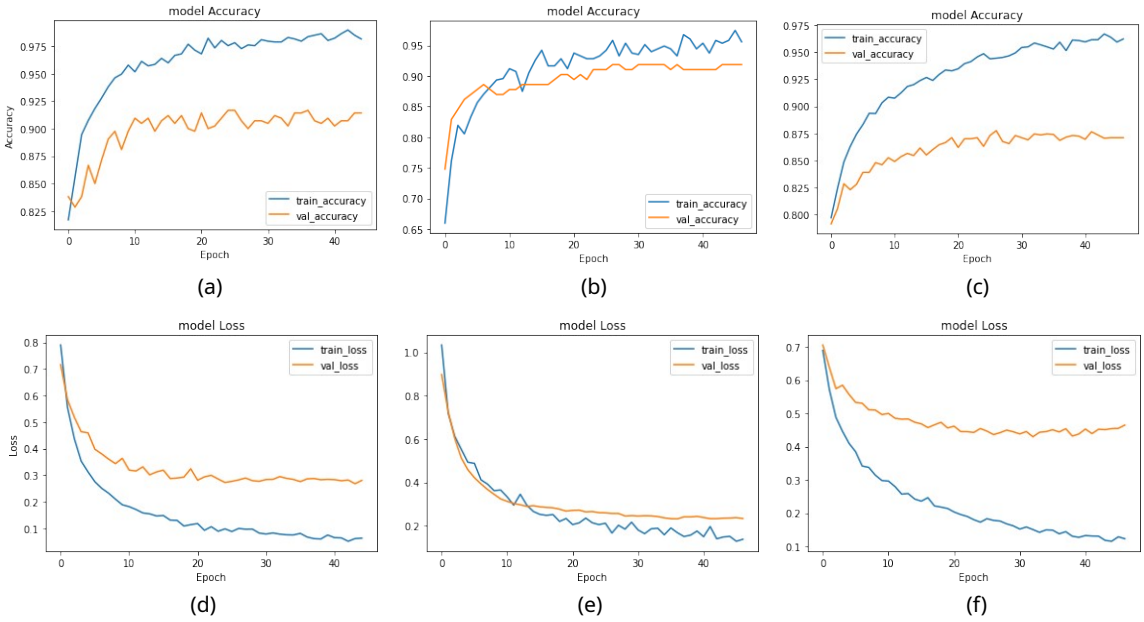


Figura 22: Desempeño del modelo de clasificación Inception-v3 mostrando la exactitud y pérdida, mientras se ejecutan las épocas en el entrenamiento. Las gráficas (a), (b) y (c) muestran la métrica de exactitud para los conjuntos de imágenes UCM, Sydney y RSICD respectivamente. Las gráficas (d), (e) y (f) muestran la métrica de pérdida para los conjuntos de datos de UCM, Sydney y RSICD respectivamente. Fuente: Elaboración propia.

3.2.3 Comportamiento del modelo Inception-v3

De manera similar a los modelos anteriores, la figura 22 presenta los valores obtenidos para las métricas de exactitud y de pérdida del modelo; conforme se ejecutan las épocas de entrenamiento correspondientes al modelo de clasificación denominado Inception-v3.

La distribución de las gráficas dentro de la figura 22, es como sigue. Las gráficas (a) y (d) corresponden al conjunto de imágenes UCM, las gráficas (b) y (e) al conjunto de imágenes Sydney y las gráficas (c) y (f) corresponden al conjunto de imágenes remotas RSICD. A su vez, las gráficas (a), (b) y (c) contienen información sobre el valor obtenido de la métrica de exactitud para cada época del entrenamiento. Por otra parte, las gráficas (d), (e) y (f) representan el valor de la pérdida obtenida durante el entrenamiento del modelo de clasificación con Inception-v3.

Ahora, sobre los resultados que se obtuvieron del entrenamiento del conjunto de imágenes remotas UCM, podemos decir lo siguiente. En la gráfica (a) se visualiza la exactitud conforme se ejecutan las épocas del modelado. Podemos observar que la ejecución termina entre las épocas 40 y 50, es decir, antes del total de las 100 épocas. Con este modelo de clasificación obtenemos muy altas proporciones de clasificación correcta, en la gráfica podemos ver como la exactitud de validación alcanza rápidamente valores alrededor del 90% de clasificación después de la época 15. La exactitud del conjunto de entrenamiento (línea azul) continúa incrementándose lentamente conforme avanzan las épocas, sin embargo, el espacio entre la línea naranja del conjunto de validación y la línea azul del conjunto de entrenamiento, se mantiene estrecho y caminando en paralelo.

En la gráfica (d), visualizamos los valores de la pérdida del modelo durante su ejecución. Observamos la pérdida del conjunto de entrenamiento (línea azul) y como continúa disminuyendo durante la ejecución de todo el experimento, de manera rápida durante las primeras épocas y con una reducción menor durante las épocas finales del entrenamiento. En esta misma gráfica (d), observamos la pérdida del conjunto de validación (línea naranja), que tiene un comportamiento similar a la línea azul del conjunto de entrenamiento. En las épocas iniciales tiene una rápida reducción, pero después de la época 20 aproximadamente, su reducción se detiene y se estabiliza alrededor del valor 0.3 con poca variabilidad en los resultados hasta que la ejecución se detiene.

El siguiente par de gráficas a describir son las etiquetadas con las letras (b) y (e) que muestran los valores obtenidos de la exactitud y la pérdida al realizar la tarea de clasificación usando el conjunto de imágenes remotas de Sydney. Este conjunto de imágenes tiene la característica de contar con solo 7 clases de imágenes, una cantidad menor que los otros dos conjuntos de imágenes, y un desbalance menor entre las clases mencionadas. En la gráfica (b) tenemos

representada a la exactitud, tanto del conjunto de entrenamiento (línea azul) como del conjunto de validación (línea naranja).

De igual manera que para el conjunto de datos anterior, vemos que la ejecución de la clasificación se detiene entre las épocas 40 y 50, cuando ya no se tiene mejora en la exactitud de validación. Podemos observar cómo la exactitud de entrenamiento continúa en aumento durante todas las épocas del experimento, con alta variabilidad en los resultados, hasta alcanzar valores por arriba del 95% de exactitud de entrenamiento. De manera similar vemos como la exactitud de validación, tiene un comportamiento similar incrementando su valor rápidamente en las épocas iniciales y después de la época 20, detener su crecimiento un poco arriba del 90%. Hacemos notar que la diferencia existente entre la exactitud de entrenamiento y la de validación para este conjunto de datos Sydney es la más reducida, al compararla con los otros ocho experimentos. Es decir, el espacio existente entre la línea azul y la naranja es el más estrecho de todos los experimentos.

Para la gráfica (e), tenemos el comportamiento de la pérdida para el conjunto de imágenes remotas Sydney, donde podemos observar que las dos líneas corren muy juntas durante la ejecución de las épocas del experimento. El modelo de clasificación Inception-v3 ha presentado un buen desempeño con el conjunto de datos Sydney. Se puede visualizar como la línea naranja del conjunto de validación, sigue muy de cerca a la línea azul del conjunto de entrenamiento. La variabilidad de los resultados para el conjunto de entrenamiento es mayor que para los resultados graficados para el conjunto de validación, es por eso que con estos valores reducidos de pérdida podemos obtener valores altos de exactitud en la ejecución de este experimento entre Inception-v3 y el conjunto de imágenes Sydney.

Continuando con la figura 22, pero ahora pasamos a la descripción de las gráficas etiquetadas con las letras (c) y (f), que corresponden a las métricas de

exactitud y pérdida respectivamente, pero ahora del conjunto de imágenes remotas denominado RSICD. Este conjunto de imágenes cuenta con 30 clases distintas, cada una conteniendo una cantidad notoriamente diferente de imágenes, lo que la hace la base de imágenes con mayor desbalance de las tres trabajadas.

La gráfica (c) nos muestra la exactitud evaluada en el conjunto de datos RSICD con el modelo de clasificación Inception-v3. Aquí observamos que la exactitud de entrenamiento sigue avanzando continuamente hasta que la ejecución del modelo se detiene, llegando a alcanzar un 96% de exactitud en el conjunto de imágenes de entrenamiento. La exactitud del conjunto de validación se incrementa en las primeras épocas de la ejecución, hasta que después de la época 20 logra una estabilización de resultados alrededor del 88%. Podemos observar en la gráfica que la ejecución del algoritmo de clasificación se detiene entre las épocas 40 y 50 debido a que ya no se presentan mejoras en los resultados. En esta gráfica también observamos que la separación entre las dos líneas de la exactitud, es la mayor cuando la comparamos contra los otros dos conjuntos de imágenes remotas.

Revisando la gráfica (f) que representa la métrica de pérdida del conjunto de imágenes remotas RSICD cuando es procesada por el modelo de clasificación Inception-v3, podemos visualizar que la pérdida de entrenamiento tiene un comportamiento hacia la baja durante toda la ejecución del modelo; hasta alcanzar valores cercanos a 0.1, dando la impresión en la imagen, que pudiera seguir bajando si se hubieran procesado más épocas. Por el contrario, la línea naranja que señala la pérdida calculada en el conjunto de validación, inicia el proceso con una reducción hasta que ya no logra continuar la reducción de su valor y se estabiliza en un valor alrededor de 0.5, después de la época 20.

3.3 Tiempos de ejecución durante el entrenamiento de los modelos de clasificación

Adicionalmente, a las métricas de desempeño que son comúnmente utilizadas durante el proceso de clasificación de imágenes; se han medido también los tiempos de ejecución de cada uno de los experimentos, es decir, la duración en segundos que tarda el algoritmo de clasificación, en procesar la cantidad total de épocas de cada experimento.

El propósito para medir los tiempos de ejecución, es porque se han utilizado dos máquinas diferentes, una de ellas posee un procesador auxiliar denominado unidad de procesamiento gráfico (GPU) y la otra no posee este procesamiento adicional y solo cuenta con la unidad central de procesamiento (CPU). La idea es hacer notar la diferencia en los tiempos de procesamiento de cada máquina y validar si existe un beneficio notable al emplear una máquina con GPU. Las características de hardware para cada tipo de máquina, se especificaron en el capítulo 2, que habla sobre la metodología.

Cada uno de los nueve experimentos sobre tiempos de ejecución se han corrido 3 veces y se ha calculado el promedio aritmético de estas tres mediciones. Los valores promedio de cada experimento para la duración del tiempo de ejecución del algoritmo de clasificación, se detallan en la Tabla 30, que muestra la duración de la ejecución cuando empleamos una máquina con procesamiento tipo GPU y en la Tabla 31, que muestra los resultados del tiempo de procesamiento cuando usamos una máquina con procesamiento tipo CPU. Como lo mencionamos previamente, los procesamientos en la máquina con GPU, se llevaron a cabo en la plataforma Google Colab y los procesamientos con máquina tipo CPU, en una computadora personal de propósito general.

Revisando los resultados de la tabla 30 podemos observar inicialmente que cuando utilizamos procesamiento con GPU, los valores para la velocidad de

procesamiento son similares para cada modelo de clasificación y conjunto de datos. Por ejemplo, la velocidad de procesamiento para el modelo VGG-16 con el conjunto de imágenes UCM es de 25.52 segundos por época, la velocidad para el modelo ResNet-50 y el conjunto UCM es de 26.28 segundos por época y la velocidad de procesamiento para el modelo Inception-v3 y el conjunto de imágenes UCM es de 25.48 segundos por época. Los demás modelos siguen este comportamiento. Es decir, los procesadores tipo GPU desempeñan una velocidad similar independientemente del modelo de clasificación usado.

Modelo	Conjunto de imágenes	Promedios para GPU		
		Duración[22]	Épocas[23]	Velocidad de procesamiento[24]
VGG-16	UCM	1199.30	47.00	25.52
	Sydney	382.17	47.33	8.07
	RSICD	7408.50	64.00	115.76
ResNet-50	UCM	1961.95	74.67	26.28
	Sydney	416.60	50.67	8.22
	RSICD	9766.74	89.33	109.33
Inception-v3	UCM	1163.65	45.67	25.48
	Sydney	374.07	49.93	7.58
	RSICD	6644.69	60.33	110.13

Tabla 30: Resultados de los tiempos de ejecución de cada uno de los modelos de clasificación y conjunto de datos, usando una máquina con procesamiento tipo GPU. Fuente: Elaboración propia.

22 Tiempo total en segundos que tarda la ejecución completa del modelo de clasificación.

23 Cantidad total de épocas que se ejecutaron durante el entrenamiento completo del modelo de clasificación.

24 Duración en segundos que le toma en promedio al modelo de clasificación ejecutar una época.

Ahora, para obtener una medida global de desempeño para cada modelo de clasificación, se ha promediado la velocidad de procesamiento de los tres conjuntos de imágenes por modelo. Los resultados obtenidos son que para el modelo de clasificación VGG-16 tenemos una velocidad de procesamiento promedio de 49.78 segundos por época, para el modelo ResNet-50 tenemos una velocidad de 47.94 segundos por época y para el modelo Inception-v3 son 47.73 segundos por época. Si los ordenamos de manera ascendente por su velocidad, tenemos que Inception-v3 es el más veloz, seguido por ResNet-50 y en tercer lugar VGG-16.

Adicionalmente, revisando los resultados globales promedio por modelo de clasificación para las mediciones del tiempo total de ejecución (Duración) y la cantidad total de épocas por corrida (Épocas), tenemos que para la duración usando GPU, el modelo VGG-16 se tarda 2,996.66 segundos totales promedio, el modelo ResNet-50 tarda 4,048.43 segundos y el modelo Inception-v3 se ejecuta en promedio en 2,727.47 segundos. Por otra parte, los resultados de la cantidad total de épocas que toma la ejecución del modelo de clasificación cuando empleamos el modelo VGG-16 es de 52.78 épocas en promedio, para el modelo ResNet-50 es de 71.56 épocas y para el modelo Inception-v3 es de 51.78 épocas. De estos resultados podemos decir que el modelo que más tiempo se toma en la ejecución es ResNet-50 con 4,048.43 segundos totales y el más rápido en su ejecución promedio es Inception-v3 con 2,727.47 segundos. Sin embargo, el modelo de clasificación ResNet-50 es el que más épocas procesa, con 71.56 épocas en promedio, VGG-16 e Inception-v3 se ejecutan en una cantidad menor de épocas promedio. De tal manera, que cuando utilizamos procesadores tipo GPU, la velocidad de procesamiento promedio permanece similar para los tres modelos de clasificación.

Adicionalmente, podemos agregar el dato del desempeño para cada uno de los modelos de clasificación y tener una idea más completa de cómo se afecta

este indicador con los valores de tiempos de ejecución y épocas procesadas. Así, podemos mencionar a uno de los indicadores del desempeño del modelo (ver tabla 27) por ejemplo, f1-score que nos calcula el promedio armónico entre la precisión y el recall para cada modelo de clasificación. En dicha tabla 27 podemos ver que el resultado del desempeño con el indicador f1-score para el modelo VGG-16 es de 0.9848, para el modelo ResNet-50 es de 0.8401 y para el modelo de clasificación Inception-v3 es de 0.9657.

De esta manera, observamos que el modelo de clasificación VGG-16 es el que mejor desempeña en f1-score, pero ocupa el segundo lugar cuando medidos el tiempo de ejecución del algoritmo de clasificación. Continuando con la métrica de desempeño, tenemos que la segunda posición la ocupa el modelo Inception-v3 y cuando hablamos de la duración de la ejecución del algoritmo de clasificación este modelo ocupa la primera posición entre los tres.

Sobre los tiempos de duración que cada modelo presenta cuando se ejecuta el algoritmo para cada una de las tres bases de imágenes remotas, vemos que los tiempos varían grandemente. Sin embargo, tenemos que consistentemente para los tres modelos de clasificación, el conjunto de imágenes remotas Sydney es el que menor tiempo de ejecución se toma, en segundo lugar aparece el conjunto de imágenes UCM y el conjunto que más tiempo se toma en la ejecución es RSICD. Entonces inferimos que la duración de la ejecución del algoritmo de procesamiento está relacionada con la cantidad de imágenes que cada conjunto contiene, es decir, a mayor cantidad de imágenes de la base, entonces mayor tiempo de ejecución.

Ahora, tomando como base al conjunto de imágenes remotas y su asociación con cada modelo de clasificación, tenemos los siguientes comentarios a los resultados encontrados. Al procesar el conjunto de imágenes UCM empleando procesadores del tipo GPU, tenemos que el modelo de clasificación VGG-16 se toma 25.52 segundos por época en promedio cuando procesa toda la

base de imágenes, ResNet-50 se tarda 26.28 segundos por época y el modelo Inception-v3 se tarda en ejecutar 25.48 segundos por época. Por lo tanto, en promedio Inception-v3 es el más veloz al momento de ejecutar el conjunto de imágenes de UCM.

Cuando los modelos de clasificación procesan el conjunto de imágenes remotas Sydney, tenemos que VGG-16 se tarda 8.07 segundos por época durante el procesamiento de la base completa, ResNet-50 se toma 8.22 segundos por época e Inception-v3 se ejecuta en 7.58 segundos por época; por los datos anteriores vemos que Inception-v3 es de nuevo el más veloz al procesar la base de imágenes remotas Sydney, seguido muy de cerca por VGG-16.

El procesamiento de la base de imágenes RSICD, la más grande de las tres, le toma a VGG-16 115.76 segundos por época, al modelo ResNet-50 le toma 109.33 segundos por época y al modelo Inception-v3 le lleva un promedio de 110.13 segundos por época. De los resultados para el conjunto de imágenes remotas RSICD, tenemos que ResNet-50 es el más veloz, seguido de cerca por Inception-v3 y cinco segundos después viene el modelo VGG-16.

Modelo	Conjunto de imágenes	Promedios para CPU		
		Duración	Épocas	Velocidad de procesamiento
VGG-16	UCM	6455.02	56.67	113.91
	Sydney	1821.88	47.33	38.49
	RSICD	33573.51	61.33	547.39
ResNet-50	UCM	3790.34	78.67	48.18
	Sydney	709.51	46.67	15.20
	RSICD	22737.79	100.00	227.38
Inception-v3	UCM	2080.95	83.00	25.07
	Sydney	408.83	55.00	7.43
	RSICD	5746.08	50.33	114.16

Tabla 31: Resultados de los tiempos de ejecución de cada uno de los modelos de clasificación y conjunto de datos, cuando usamos una máquina con unidad central de procesamiento (CPU). Fuente: Elaboración propia.

Revisando los resultados de la tabla 31 podemos observar de entrada que cuando utilizamos procesamiento con CPU, los valores de la velocidad de procesamiento se comportan diferentes para cada modelo de clasificación y cuando son comparados por conjunto de datos. Por ejemplo, la velocidad de procesamiento para el modelo VGG-16 con el conjunto de imágenes UCM es de 113.91 segundos por época, la velocidad para el modelo ResNet-50 y el conjunto UCM es de 48.18 segundos por época y la velocidad de procesamiento para el modelo Inception-v3 y el conjunto de imágenes UCM es de 25.07 segundos por época. Si comparamos estos resultados vemos diferencias notorias en la velocidad de procesamiento. Continuando con esta comparativa entre modelos de clasificación y conjunto de datos, vemos que siguen este mismo comportamiento de diferencias notables en

la velocidad de procesamiento. Es decir, los procesadores tipo CPU desempeñan una velocidad distinta según el modelo de clasificación usado.

Ya anteriormente se obtuvo una medida global de desempeño para cada modelo de clasificación, que volvemos a aplicar pero ahora a los resultados del procesamiento con CPU. Los resultados obtenidos para el modelo de clasificación VGG-16 nos proporcionan una velocidad de procesamiento promedio de 233.27 segundos por época, para el modelo ResNet-50 tenemos una velocidad de 96.92 segundos por época y para el modelo Inception-v3 son 48.89 segundos por época. Al ordenarlos de manera ascendente por su velocidad, tenemos que Inception-v3 es el más veloz, seguido por ResNet-50 y en tercer lugar VGG-16.

Adicionalmente, revisando los resultados globales promedio por modelo de clasificación para las mediciones del tiempo total de ejecución (Duración) y la cantidad total de épocas por corrida (Épocas), tenemos que para la duración usando CPU, el modelo VGG-16 se tarda 13,950.14 segundos totales promedio, el modelo ResNet-50 tarda 9,079.21 segundos y el modelo Inception-v3 se ejecuta en promedio en 2,745.29 segundos. Por otra parte, los resultados de la cantidad total de épocas que toma la ejecución del modelo de clasificación cuando empleamos el modelo VGG-16 es de 55.11 épocas en promedio, para el modelo ResNet-50 es de 75.11 épocas y para el modelo Inception-v3 es de 48.89 épocas. De estos resultados podemos decir que el modelo que más tiempo se toma en la ejecución es VGG-16 con 13,950.14 segundos totales y el más rápido en su ejecución promedio es Inception-v3 con 2,745.29 segundos. Sin embargo, el modelo de clasificación ResNet-50 es el que más épocas procesa, con 75.11 épocas en promedio, VGG-16 se ejecuta con la menor cantidad de épocas promedio.

Sobre los tiempos de duración que cada modelo presenta cuando se ejecuta el algoritmo para cada una de las tres bases de imágenes remotas, vemos que también cuando usamos CPU, los tiempos varían grandemente. Sin embargo,

tenemos que consistentemente para los tres modelos de clasificación, el conjunto de imágenes remotas Sydney es el que menor tiempo de ejecución se toma, en segundo lugar aparece el conjunto de imágenes UCM y el conjunto que más tiempo se toma en la ejecución es RSICD. Entonces, también para los resultados cuando empleamos un tipo de procesamiento CPU, tenemos que la duración de la ejecución del algoritmo de procesamiento está relacionada con la cantidad de imágenes que cada conjunto contiene, es decir, a mayor cantidad de imágenes de la base, entonces mayor tiempo de ejecución.

Ahora, observando los resultados desde la perspectiva del conjunto de imágenes remotas y su asociación con cada modelo de clasificación, tenemos los siguientes comentarios a los resultados encontrados (ver tabla 31). Al ajustar el conjunto de imágenes UCM con procesadores del tipo CPU, tenemos que el modelo de clasificación VGG-16 se toma 113.91 segundos por época en promedio cuando procesa todas las imágenes, ResNet-50 se tarda 48.18 segundos por época y el modelo Inception-v3 se tarda en ejecutar 25.07 segundos por época. Por lo tanto, en promedio Inception-v3 es el más veloz al momento de ejecutar el conjunto de imágenes de UCM.

Cuando los modelos de clasificación procesan el conjunto de imágenes remotas Sydney, tenemos que VGG-16 se tarda 38.49 segundos por época durante el procesamiento de la base completa, ResNet-50 se toma 15.20 segundos por época e Inception-v3 se ejecuta en 7.43 segundos por época; por los datos anteriores vemos que Inception-v3 es de nuevo el más veloz al procesar la base de imágenes remotas Sydney.

El procesamiento de la base de imágenes RSICD, la más grande de las tres, le toma a VGG-16 547.39 segundos por época, al modelo ResNet-50 le toma 227.38 segundos por época y al modelo Inception-v3 le lleva un promedio de 114.16 segundos por época. De los resultados para el conjunto de imágenes

remotas RSICD, tenemos que Inception-v3 es el más veloz, seguido por ResNet-50 y mucho más atrás viene el modelo VGG-16.

3.4 Predicción de imágenes

El modelo entrenado con las imágenes de los conjuntos de datos utilizados en este trabajo nos sirve para llevar a cabo predicciones sobre imágenes remotas que nunca hayan sido procesadas por el modelo de clasificación.

Recordemos que de cada uno de los conjuntos de imágenes remotas hemos guardado el 10% de cada conjunto y estas imágenes no han sido usadas en ninguna etapa anterior del entrenamiento del modelo. A este subconjunto de imágenes le denominamos conjunto de prueba. Podemos decir, que para el modelo de clasificación, son imágenes nuevas que deseamos que sean clasificadas en una de las clases que cada conjunto maneja.

Para cada conjunto de datos se selecciona una sola imagen aleatoriamente del conjunto de prueba, de tal manera que para el conjunto de imágenes remotas UCM, se han seleccionado aleatoriamente 21 imágenes, para el conjunto Sydney se seleccionaron 7 imágenes y para el conjunto RSICD se seleccionaron 30 imágenes aleatorias.

Estas imágenes se inyectan al modelo de clasificación correspondiente previamente entrenado y nos entrega como resultado la clase que el modelo ha predicho.

En las tablas 32 y 33 se resumen los resultados de la predicción para cada modelo de clasificación y conjunto de imágenes. Hablando de manera general, las dos tablas anteriores presentan un ejercicio de los porcentajes de predicciones de cada modelo de clasificación. Los resultados presentados son para una sola muestra de un grupo de imágenes seleccionadas aleatoriamente, es decir,

podríamos seleccionar otra nueva muestra aleatoria de imágenes del conjunto de prueba y los resultados serán diferentes.

En el anexo 2 se presentan las visualizaciones de las predicciones que se han resumido en las tablas 32 y 33. Se presentan nueve visualizaciones, cuando se ha entrenado con procesamiento con GPU y otras nueve visualizaciones cuando se ha implementado con procesador CPU.

En la tabla 32 podemos ver los resultados de las predicciones de los algoritmos de clasificación cuando se procesan mediante circuitos GPU. Para cada modelo de clasificación y conjunto de imágenes remotas, tenemos la cantidad de imágenes remotas a predecir, la cantidad de imágenes que fueron correctamente clasificadas por el algoritmo y la proporción de imágenes correctas. Se agregó una columna con el resultado de la métrica de desempeño f1-score, que nos permite establecer una referencia contra el porcentaje de imágenes correctas.

Revisando los resultados para el modelo de clasificación VGG-16, vemos que el modelo para el conjunto de imágenes Sydney alcanzó el 100% de imágenes correctas en este ejercicio puntual, recordemos que el conjunto Sydney tiene solo 7 imágenes para predecir porque únicamente contiene 7 clases. En seguida viene UCM que alcanzó el 85.71% y no muy atrás está el conjunto RSICD con 83.33%. Estos resultados son congruentes con los resultados de desempeño obtenidos por el algoritmo de clasificación durante su entrenamiento.

Modelo	Conjunto de imágenes	GPU			
		Total de imágenes	Imágenes correctas	Porcentaje de correctas	Métrica f1-score
VGG-16	UCM	21	18	85.71%	98.92%
	Sydney	7	7	100%	98.65%
	RSICD	30	25	83.33%	97.88%
ResNet-50	UCM	21	14	66.67%	87.51%
	Sydney	7	7	100%	95.42%
	RSICD	30	15	50%	69.09%
Inception-v3	UCM	21	21	100%	98.02%
	Sydney	7	6	85.71%	95.54%
	RSICD	30	24	80%	96.15%

Tabla 32: Ejercicio de predicción de imágenes sobre un grupo aleatorio del conjunto de prueba. Se presenta el porcentaje de imágenes correctas por modelo y base de imágenes. Fuente: Elaboración propia.

Sobre los resultados de predicciones para el algoritmo ResNet-50, podemos confirmar que los resultados de predicción para los conjuntos de imágenes de UCM y RSICD, son los más bajos de los 3 modelos de clasificación. El modelo ResNet-50 obtiene un porcentaje de predicción puntual de 66.67% cuando se le pide predecir imágenes del conjunto de datos de UCM. Pero este mismo algoritmo funciona muy bien cuando realiza predicciones sobre el conjunto de imágenes Sydney, otorgando un 100% de exactitud. Sin embargo, cuando ResNet-50 ejecuta las predicciones sobre las imágenes de prueba del conjunto RSICD, obtiene el porcentaje más bajo de la tabla, con 50% de imágenes correctas. Este resultado bajo, también está en línea con los resultados de desempeño para este modelo y conjunto de imágenes.

Los resultados de predicciones puntuales para el modelo Inception-v3, de la tabla 32, presentan una exactitud excelente para el conjunto UCM, prediciendo el 100% de las imágenes de prueba. Este modelo predijo el 85.71% de las imágenes del conjunto Sydney y el resultado de predicción cuando trabajó con imágenes del conjunto RSICD fue del 80%.

Se obtuvo el porcentaje de predicción por modelo de clasificación, al obtener la proporción de la suma de imágenes correctas entre la suma de imágenes totales. Así, usando los valores de la tabla 32 (GPU), tenemos que el modelo de clasificación VGG-16, nos entrega un porcentaje de predicción de 86.21%; el modelo ResNet-50 nos entrega 62.07% de certeza en la predicción y el modelo Inception-v3 nos da como resultado un 87.93% en el ejercicio de predicción cuando sumamos los tres conjuntos de datos.

De manera análoga, en la tabla 33, se han registrado los resultados de un ejercicio de predicción para los modelos de clasificación ejecutados con una máquina que cuenta con procesador tipo CPU. Los resultados se presentan de manera similar a la tabla con procesamiento tipo GPU, se ha obtenido la proporción de imágenes correctas sobre el total de imágenes.

Podemos observar en la tabla 33, que el porcentaje de predicción para el modelo VGG-16 con el conjunto de imágenes remotas UCM, es del 95.24%; y está muy por arriba de los porcentajes de predicción resultantes cuando utilizamos los conjuntos de datos Sydney y RSICD, cuyos resultados son de 71.43% y 73.33% respectivamente.

Modelo	Conjunto de imágenes	CPU			
		Total de imágenes	Imágenes correctas	Porcentaje de correc-tas	Métrica f1-score
VGG-16	UCM	21	20	95.24%	98.92%
	Sydney	7	5	71.43%	98.65%
	RSICD	30	22	73.33%	97.88%
ResNet-50	UCM	21	14	66.67%	87.51%
	Sydney	7	6	85.71%	95.42%
	RSICD	30	15	50%	69.09%
Inception-v3	UCM	21	20	95.24%	98.02%
	Sydney	7	7	100%	95.54%
	RSICD	30	26	86.67%	96.15%

Tabla 33: Ejercicio de predicción de imágenes sobre un grupo aleatorio del conjunto de prueba cuando empleamos procesamiento con CPU. Se presenta el porcentaje de imágenes correctas por modelo y base de imágenes. Fuente: Elaboración propia.

El modelo de clasificación ResNet-50 tiene su mejor resultado de predicción con el conjunto de imágenes Sydney, alcanzando un 85.71% en el ejercicio. El segundo mejor porcentaje para el modelo ResNet-50 es para el conjunto de imágenes UCM con un 66.67% de predicción real y luego tenemos al conjunto de imágenes RSICD que alcanzó solo un 50% de predicciones reales durante el ejercicio. La predicción del modelo ResNet-50 combinado con el conjunto de imágenes remotas RSICD es el más bajo de todos los ejercicios de predicción ejecutados con procesador tipo CPU. Y en general el modelo ResNet-50 es el que presenta los resultados de predicción más bajos en la tabla 33.

Para el modelo de clasificación Inception-v3, tenemos que el mejor resultado de predicción se obtuvo cuando se combinó con el conjunto Sydney con

un 100% de imágenes correctas. Seguido muy de cerca por el resultado del conjunto de imágenes UCM con un 95.24% de predicciones correctas y posteriormente tenemos al conjunto de imágenes remotas RSICD con un 86.67% de predicciones correctas.

Como información adicional, se obtuvo el porcentaje de predicción global por modelo de clasificación condensando los resultados de cada conjunto de datos y presentando un solo resultado por modelo. Así, tenemos que los mejores resultados de predicción para este ejercicio fueron para el modelo Inception-v3 que alcanzó un 91.38% de predicción de imágenes correctas; seguido por el modelo VGG-16 que obtuvo un 81.03% de imágenes correctas y más abajo se ubicaron los resultados de predicción del modelo de clasificación ResNet-50 que logró un 60.34% de predicciones correctas.

Conclusiones

Conclusiones

A través de la implementación de las distintas etapas del presente proyecto, se han presentado las diferencias existentes entre varios modelos de clasificación basados en algoritmos de redes neuronales artificiales. Y el proceso de clasificación se ha llevado a cabo sobre tres conjuntos de imágenes remotas seleccionadas para este propósito. El propósito general del proyecto es presentar los resultados diferenciadores de cada modelo, cuando medimos el desempeño, el tiempo de ejecución y el poder de generalización de cada uno de ellos.

La clasificación se llevó a cabo sobre imágenes remotas, que son fotografías de alta resolución, tomadas desde gran altura y que tienen características y funciones diferentes a las fotografías de personas y lugares comunes. Este campo de clasificación de imágenes remotas es novedoso y presenta varias utilidades como el conocimiento del uso de suelo y la clasificación de tipos de sembradíos.

Se obtuvieron tres conjuntos de imágenes remotas y se pudo hacer uso de ellas de manera gratuita. Estos tres conjuntos de imágenes remotas poseen características muy distintas entre cada uno de ellos. De tal forma, que son útiles para poner a prueba el poder de clasificación de cada uno de los modelos bajo condiciones distintas. Es decir, el conjunto de imágenes Sydney posee solo siete clases de imágenes y cuenta con un total de 613 fotografías, pero el total de imágenes no está proporcionalmente distribuido entre las siete clases, lo que lo hace un conjunto desbalanceado. El conjunto de imágenes UCM, posee 2100 imágenes distribuidas en 21 clases y cada clase tiene 100 imágenes de alguna sección de la superficie terrestre. Por estas características, UCM es un conjunto con una buena cantidad de imágenes para el ejercicio de clasificación, pero además sus clases están balanceadas, lo que hace que cada clase tenga las

mismas oportunidades de ser aprendida. El tercer conjunto de imágenes remotas es RSICD, que es un conjunto de imágenes muy grande, con más de 10,000 fotografías. Pero presenta la característica que cada una de sus 30 clases están muy desbalanceadas, la cantidad de imágenes en algunas clases es proporcionalmente menor que en otras. Entonces usando estos tres conjuntos de imágenes con características diferentes, podemos someter un mismo modelo de clasificación a circunstancias diferentes y medir su desempeño al momento de llevar a cabo la tarea de clasificación.

Se emplearon tres modelos de clasificación que funcionan con redes neuronales artificiales. Los resultados que estos modelos entregan, son de los mejores dentro de la rama de clasificación de imágenes, superando a los algoritmos predecesores. La selección de estos tres modelos en específico se basa en las diferencias entre ellos; ya que uno de ellos contiene una menor cantidad de capas, mientras que los otros dos contienen una mayor cantidad de capas, pero no están estructuradas linealmente, sino en una configuración que caracteriza a cada una de dichas redes neuronales.

Comentando sobre los modelos de clasificación y su desempeño al momento de ajustar un modelo específico, se pudo observar que el mejor modelo a través de un promedio de todas las métricas de desempeño, es VGG-16 con 98.4%, seguido muy de cerca por el modelo Inception-v3 con 96.6%. Podemos decir que estos dos modelos tienen un desempeño superior superando el 90% en todas las métricas de desempeño. En tercer lugar, aparece el modelo de clasificación ResNet-50 con un 84.5% de desempeño global en todas las métricas de desempeño, y esto se debe a un pobre resultado del modelo cuando realiza el entrenamiento de los dos conjuntos de imágenes más grandes, UCM y RSICD. Los modelos VGG-16 e Inception-v3 se desempeñan bien para los tres conjuntos de imágenes remotas, sin importar su tamaño o complejidad.

Cambiando el enfoque y ahora observando el desempeño global de cada uno de los conjuntos de imágenes remotas, tenemos el conjunto Sydney obtiene un desempeño promedio superior (arriba de 90%) para los tres modelos de clasificación utilizados. El conjunto UCM obtiene también un desempeño promedio superior y después tenemos al conjunto de imágenes RSICD con un resultado promedio de 88.3%.

Desde el punto de vista del desbalance de los conjuntos de datos y su desempeño de clasificación, tenemos que UCM es un conjunto balanceado porque todas sus 21 clases contienen 100 imágenes remotas. Es decir, la variabilidad de la cantidad de imágenes por clase, medida en desviaciones estándar, es de cero porque todas las clases contienen la misma cantidad de imágenes. El conjunto de datos Sydney presenta 7 clases con una cantidad variable de imágenes por clase. Recordemos que en total son 613 imágenes remotas en todo el conjunto de datos Sydney; con un promedio de imágenes por clase de 87.5 y una desviación estándar de 72.9 imágenes. El conjunto de datos RSICD contiene 10,921 imágenes remotas, con una cantidad variable de imágenes por clase; el promedio de imágenes por clase es de 364, mientras que la desviación estándar es de 140.25 imágenes por clase.

De las estadísticas anteriores podemos expresar que el conjunto con el mayor desbalance es RSICD, seguido por Sydney y posteriormente se encuentra UCM con cero desbalance.

Desde el enfoque del desbalance de los conjuntos de datos y su desempeño en la tarea de clasificación, se menciona que el conjunto Sydney presenta el mejor desempeño, tomando como referencia la métrica f1 score y promediando los valores de los tres modelos de clasificación, con 96.53%; seguido de cerca por el conjunto UCM, que nos muestra un 94.82% de desempeño en f1 score; y más atrás en el desempeño encontramos al conjunto de imágenes RSICD

con una medida de 87.71%. Entonces desde el punto de vista del desbalance, podemos comentar que aquellos conjuntos con una menor medida de variabilidad, son los que presentan mejor desempeño. Además, considerando los resultados el desempeño se puede mantener, aún en presencia de variabilidad, si reducimos la cantidad de clases dentro de nuestro conjunto de imágenes remotas. Tal es el caso para los conjuntos UCM y Sydney, que presentan mejores resultados de desempeño, cuando los comparamos contra el conjunto de datos RSICD.

Vemos que los conjuntos de imágenes Sydney y UCM, alcanzan desempeños promedio por arriba de 95%. El conjunto Sydney mantiene el desempeño superior para los tres modelos de clasificación, a diferencia del conjunto UCM, que presenta un desempeño arriba del 90% para los modelos VGG-16 e Inception-v3, pero cuando se trabaja con el modelo ResNet-50 no se obtiene el desempeño superior antes mencionado. Sobre el conjunto de imágenes RSICD, también mantiene un desempeño por arriba del 90% para los modelos VGG-16 e Inception-v3, pero con el modelo ResNet-50 presenta el desempeño más bajo de todos los nueve experimentos realizados.

Entonces, el conjunto de imágenes RSICD, debido a su gran tamaño en cantidad de imágenes y además por su gran desbalance entre clases, es el que obtiene los resultados de desempeño más bajos entre los tres modelos de clasificación y en específico con el modelo ResNet-50. En contraparte, los mejores resultados de clasificación se obtienen cuando usamos un conjunto de datos balanceado como UCM; aunque los resultados no sean muy distintos para las otras dos bases de datos.

Además de las comparativas de desempeño en la tarea de clasificación, se llevó a cabo la medición del tiempo de ejecución para cada tipo de hardware de procesamiento. De tal manera, que se analizaron los resultados del tiempo que

tardan las combinaciones de algoritmos y conjunto de datos, cuando procesan los pasos en la etapa de clasificación.

La primera conclusión sobre las velocidades de ejecución durante el proceso de clasificación, están directamente relacionados con el tamaño de la base de datos, tanto cuando empleamos CPU como cuando empleamos GPU. Las bases de imágenes más pequeñas se ejecutan en un menor tiempo total.

La cantidad de épocas de cada proceso de clasificación, se relaciona más apropiadamente con las imágenes seleccionadas en cada lote aleatorio y con los parámetros de los experimentos que con la complejidad de la base de imágenes remotas.

Sobre las velocidades de ejecución promedio cuando trabajamos con procesador tipo CPU en específico, tenemos que el modelo más rápido es Inception-v3 con 48.89 segundos por época. En segundo lugar, tenemos a ResNet-50 que tarda en promedio casi el doble que Inception-v3, con 96.92 segundos por época y el más lento en este tipo de procesador es VGG-16 que tarda 233.27 segundos por época, es decir, tarda más de cuatro veces lo que tarda Inception-v3.

Por otra parte, hablando del procesador tipo GPU, tenemos que los tres modelos de clasificación se ejecutan en velocidades promedio muy similares, sin importar el tamaño de la base de datos ni el modelo de clasificación. Así, tenemos que VGG-16 se toma 49.78 segundos por época, ResNet-50 toma en promedio 47.94 segundos por época e Inception-v3 tarda 47.73 segundos por época. Con este tipo de procesador GPU (usando la plataforma Google Colab), existe una relación entre el tiempo total de ejecución y la cantidad de épocas por corrida, es decir, a mayor tiempo de ejecución, mayor cantidad de épocas por corrida de clasificación.

En términos generales, el utilizar procesadores tipo GPU, nos reduce cuando menos en un 100% los tiempos de ejecución del algoritmo. Por lo que en general es una mejor opción que aquellas máquinas con procesadores tipo CPU. La excepción a esta propuesta es el modelo Inception-v3, que tiene tiempos y velocidades de ejecución muy similares tanto en CPU como en GPU. Lo que lo hace un modelo de clasificación de elección cuando no contamos con una máquina que incluya un procesador tipo GPU.

Por otra parte, una vez que se ha entrenado el modelo de clasificación, este nos sirve para llevar a cabo ejercicios de predicción sobre conjuntos de imágenes remotas que nunca antes han sido reconocidas por el algoritmo. Mediante estos ejercicios de predicción, ponemos a prueba el poder de generalización del modelo obtenido, así como los parámetros específicos seleccionados.

Pudimos observar con los resultados obtenidos en los ejercicios de predicción de imágenes de prueba, que se mantienen los desempeños observados durante la etapa de entrenamiento y validación de los modelos de clasificación. Es decir, al momento de evaluar nuevas imágenes remotas, el modelo Inception-v3 se mantiene como aquel que entrega los mejores resultados de predicción. Luego tenemos al modelo VGG-16, con resultados muy cercanos al primero. El modelo ResNet-50 presentó un desempeño bajo al momento de realizar los ejercicios de predicción. Entonces bajo estos resultados obtenidos por los modelos configurados, podemos decir que los modelos Inception-v3 y VGG-16 son los que nos entregaron los mejores resultados de exactitud, al momento de llevar a cabo predicciones de imágenes remotas.

Basados únicamente en los resultados obtenidos durante la ejecución de los ejercicios del presente proyecto, tenemos que el modelo de clasificación Inception-v3 es el que mejor desempeño general tiene, no solo cuando lo medimos con la métrica de exactitud, sino también cuando utilizamos las métricas

de precisión, recall y f1-score. Se desempeña bien a través de los tres conjuntos de imágenes remotas utilizados, sin importar la cantidad de imágenes ni el desbalance entre clases de cada conjunto. Además, tiene la característica de presentar un muy buen desempeño aun cuando se ejecutan los algoritmos usando procesador tipo CPU. Esto lo hace un excelente modelo de clasificación para usarse en diversas condiciones.

El modelo de clasificación VGG-16 también maneja valores de desempeño similares a Inception-v3 cuando lo medimos empleando las métricas de exactitud, precisión, recall y f1-score. Sin embargo, se ejecuta en un mayor tiempo y cuando se somete al entrenamiento con procesador tipo CPU, toma un tiempo mucho mayor que usando GPU.

Se pudo obtener de manera gratuita, tres conjuntos de imágenes remotas que reunían las características necesarias para los experimentos. Al combinar estos conjuntos de imágenes, los modelos de clasificación se entrenaban con bases de imágenes de diferente tamaño, con diferente cantidad de clases y también con distintas cantidades de imágenes por clase. Todas estas diferencias entre conjunto de imágenes fueron con la finalidad de aumentar el poder de generalización en la predicción de los modelos entrenados durante los experimentos.

La rama del análisis e interpretación de imágenes remotas es muy amplia, y después de concluir el presente análisis comparativo de modelos de clasificación, podemos mencionar como trabajos futuros el ampliar el análisis de clasificación de imágenes remotas usando modelos basados en aprendizaje de máquina como por ejemplo, máquinas de soporte vectorial (SVM, por sus siglas en inglés) o modelos alternativos de redes neuronales como las máquinas Boltzmann restringidas (RBM, por sus siglas en inglés); y realizar comparativas de los resultados obtenidos con modelos que difieren conceptualmente.

Otro aspecto que ha quedado pendiente para trabajos futuros es el emplear conjuntos de datos diferentes que contengan una gran cantidad de imágenes remotas pero que el balance entre clases sea similar, de tal manera que el análisis de desempeño del modelo se lleve a cabo por los diferentes temas que maneje cada conjunto de datos, como por ejemplo imágenes de tipo de siembras o el análisis de áreas lacustres.

Adicional a los temas anteriores, podemos mencionar como trabajo pendiente el proceso automático de descripción de imágenes remotas; que consideramos como el paso siguiente a la clasificación de imágenes o escenas. Se trata de la interpretación de lo que se observa en la imagen completa y es plasmado en un texto que acompaña a la imagen y describe lo que se observa en dicha imagen. Todo esto desde luego ejecutado por una máquina de manera automática.

Bibliografía

[1] Planetek Italia, “The History of Remote Sensing”, Planetek Italia. Consultado: el 2 de octubre de 2023. [En línea]. Disponible en: https://www.planetek.it/eng/training_courses/online_manuals/on_line_course_of_remote_sensing/2_the_history_of_remote_sensing

[2] G. K. MOORE, “What is a picture worth? A history of remote sensing / Quelle est la valeur d’une image? Un tour d’horizon de télédétection”, *Hydrol. Sci. Bull.*, vol. 24, núm. 4, pp. 477–485, dic. 1979, doi: 10.1080/02626667909491887.

[3] “What is remote sensing and what is it used for? | U.S. Geological Survey”. Consultado: el 3 de octubre de 2023. [En línea]. Disponible en: https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used#publications

[4] J. Kalajdjieski *et al.*, “Air Pollution Prediction with Multi-Modal Data and Deep Neural Networks”, *Remote Sens.*, vol. 12, núm. 24, Art. núm. 24, ene. 2020, doi: 10.3390/rs12244142.

[5] S. Dhingra y D. Kumar, “A review of remotely sensed satellite image classification”, *Int. J. Electr. Comput. Eng. IJECE*, vol. 9, núm. 3, p. 1720, jun. 2019, doi: 10.11591/ijece.v9i3.pp1720-1731.

[6] R. Szeliski, “What is computer vision?”, en *Computer Vision: Algorithms and Applications*, 2nd Edition., 2021, pp. 1–32.

[7] S. A. Medjahed, “A Comparative Study of Feature Extraction Methods in Images Classification”, *Int. J. Image Graph. Signal Process.*, vol. 7, pp. 16–23, feb. 2015, doi: 10.5815/ijigsp.2015.03.03.

[8] Y. Yang y S. Newsam, “Bag-of-visual-words and spatial extensions for land-use classification”, en *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, en GIS ’10. New York, NY, USA: Association for Computing Machinery, nov. 2010, pp. 270–279. doi: 10.1145/1869790.1869829.

[9] F. Zhang, B. Du, y L. Zhang, “Saliency-Guided Unsupervised Feature Learning for Scene Classification”, *Geosci. Remote Sens. IEEE Trans. On*, vol. 53, mar. 2015, doi: 10.1109/TGRS.2014.2357078.

[10] X. Lu, B. Wang, X. Zheng, y X. Li, “Exploring Models and Data for Remote Sensing Image Caption Generation”, *IEEE Trans. Geosci. Remote Sens.*, vol. 56, núm. 4, pp. 2183–2195, abr. 2018, doi: 10.1109/TGRS.2017.2776321.

[11] J. Song, S. Gao, Y. Zhu, y C. Ma, “A survey of remote sensing image classification based on CNNs”, *Big Earth Data*, vol. 3, núm. 3, pp. 232–254, jul. 2019, doi: 10.1080/20964471.2019.1657720.

[12] D. J. Hand, “Assessing the Performance of Classification Methods”, *Int. Stat. Rev.*, vol. 80, núm. 3, pp. 400–414, 2012, doi: 10.1111/j.1751-5823.2012.00183.x.
[13] G. Cheng y J. Han, “A survey on object detection in optical remote sensing images”, *ISPRS J. Photogramm. Remote Sens.*, vol. 117, pp. 11–28, jul. 2016, doi: 10.1016/j.isprsjprs.2016.03.014.
[14] B. Zhao, “A Systematic Survey of Remote Sensing Image Captioning”, *IEEE Access*, vol. 9, pp. 154086–154111, 2021, doi: 10.1109/ACCESS.2021.3128140.
[15] L. Baischer, M. Wess, y N. TaheriNejad, “Learning on Hardware: A Tutorial on Neural Network Accelerators and Co-Processors”. arXiv, el 19 de abril de 2021. doi: 10.48550/arXiv.2104.09252.
[16] “What is Computer Vision? | IBM”. Consultado: el 30 de mayo de 2023. [En línea]. Disponible en: https://www.ibm.com/topics/computer-vision
[17] “MNIST Demos on Yann LeCun’s website”. Consultado: el 30 de mayo de 2023. [En línea]. Disponible en: http://yann.lecun.com/exdb/lenet/
[18] M. Li, S. Zang, B. Zhang, S. Li, y C. Wu, “A Review of Remote Sensing Image Classification Techniques: the Role of Spatio-contextual Information”, *Eur. J. Remote Sens.*, vol. 47, núm. 1, pp. 389–411, ene. 2014, doi: 10.5721/EuJRS20144723.
[19] J. F. Mas y J. J. Flores, “The application of artificial neural networks to the analysis of remotely sensed data”, *Int. J. Remote Sens.*, vol. 29, núm. 3, pp. 617–663, feb. 2008, doi: 10.1080/01431160701352154.
[20] T. Lillesand, R. W. Kiefer, y J. Chipman, *Remote Sensing and Image Interpretation*. John Wiley & Sons, 2015.
[21] M. Salah, “A survey of modern classification techniques in remote sensing for improved image classification”, vol. 11, núm. 1, p. 21, 2017.
[22] D. Lu y Q. Weng, “A survey of image classification methods and techniques for improving classification performance”, *Int. J. Remote Sens.*, vol. 28, núm. 5, pp. 823–870, mar. 2007, doi: 10.1080/01431160600746456.
[23] D. Chen y D. Stow, “The Effect of Training Strategies on Supervised Classification at Different Spatial Resolutions”, *Photogramm. Eng.*, nov. 2002.
[24] X. Hao, L. Liu, R. Yang, L. Yin, L. Zhang, y X. Li, “A Review of Data Augmentation Methods of Remote Sensing Image Target Recognition”, *Remote Sens.*, vol. 15, núm. 3, Art. núm. 3, ene. 2023, doi: 10.3390/rs15030827.
[25] P. Stanchev, D. Green, y B. Dimitrov, “High level color similarity retrieval”, *Int. J.*, vol. 10, ene. 2003.
[26] T. Dong P., “A Review on Image Feature Extraction and Representation Techniques”, jul. 2013, Consultado: el 14 de junio de 2023. [En línea]. Disponible en: https://gvpress.com/journals/IJMUE/vol8_no4/39.pdf

[27] D. Zhang y G. Lu, “Review of shape representation and description techniques”, *Pattern Recognit.*, vol. 37, núm. 1, pp. 1–19, ene. 2004, doi: 10.1016/j.patcog.2003.07.008.
[28] T. C. Lei, S. Wan, y T. Y. Chou, “The comparison of PCA and discrete rough set for feature extraction of remote sensing image classification – A case study on rice classification, Taiwan”, *Comput. Geosci.*, vol. 12, núm. 1, pp. 1–14, mar. 2008, doi: 10.1007/s10596-007-9057-7.
[29] D. Lu y Q. Weng, “Spectral Mixture Analysis of the Urban Landscape in Indianapolis with Landsat ETM+ Imagery”, *Photogramm. Eng. Remote Sens.*, vol. 70, núm. 9, pp. 1053–1062, sep. 2004, doi: 10.14358/PERS.70.9.1053.
[30] W. L. Stefanov, M. S. Ramsey, y P. R. Christensen, “Monitoring urban land cover change: An expert system approach to land cover classification of semiarid to arid urban centers”, *Remote Sens. Environ.*, vol. 77, núm. 2, pp. 173–185, ago. 2001, doi: 10.1016/S0034-4257(01)00204-8.
[31] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, y Y. Miao, “Review of Image Classification Algorithms Based on Convolutional Neural Networks”, *Remote Sens.*, vol. 13, núm. 22, Art. núm. 22, ene. 2021, doi: 10.3390/rs13224712.
[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, y Z. Wojna, “Rethinking the Inception Architecture for Computer Vision”, presentado en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826. Consultado: el 17 de noviembre de 2022. [En línea]. Disponible en: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html
[33] M. S. Navin y L. Agilandeeswari, “Comprehensive review on land use/land cover change classification in remote sensing”, *J. Spectr. Imaging*, vol. 9, jul. 2020, doi: 10.1255/jsi.2020.a8.
[34] M. Digra, R. Dhir, y N. Sharma, “Land use land cover classification of remote sensing images based on the deep learning approaches: a statistical analysis and review”, *Arab. J. Geosci.*, vol. 15, núm. 10, p. 1003, may 2022, doi: 10.1007/s12517-022-10246-8.
[35] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, y S. Homayouni, “Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review”, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 6308–6325, 2020, doi: 10.1109/JSTARS.2020.3026724.
[36] M. Govender, K. Chetty, y H. Bulcock, “A review of hyperspectral remote sensing and its application in vegetation and water resource studies”, *Water SA*, vol. 33, núm. 2, pp. 145–151, abr. 2007, doi: 10.10520/EJC116430.
[37] Y. Wang, X. Zhu, y B. Wu, “Automatic detection of individual oil palm trees from UAV images using HOG features and an SVM classifier”, *Int. J. Remote*

Sens., vol. 40, núm. 19, pp. 7356–7370, oct. 2019, doi: 10.1080/01431161.2018.1513669.
[38] W. Liang, L. Yijian, C. Haiyan, Jiangpeng, yin Wenxin, y W. Weikang, "Combining UAV-Based Vegetation Indices and Image Classification to Estimate Flower Number in Oilseed Rape". Consultado: el 27 de noviembre de 2023. [En línea]. Disponible en: https://www.mdpi.com/2072-4292/10/9/1484
[39] Nasirun Mohd. Saleh, "Mapping atmospheric pollution using remote sensing". Consultado: el 27 de agosto de 2022. [En línea]. Disponible en: https://spie.org/news/0853-mapping-atmospheric-pollution-using-remote-sensing?SSO=1
[40] Y. Xie, Z. Sha, y M. Yu, "Remote sensing imagery in vegetation mapping: a review", *J. Plant Ecol.*, vol. 1, núm. 1, pp. 9–23, mar. 2008, doi: 10.1093/jpe/rtm005.
[41] W. Lv y X. Wang, "Overview of Hyperspectral Image Classification", *J. Sens.*, vol. 2020, p. e4817234, jul. 2020, doi: 10.1155/2020/4817234.
[42] C.-C. Yang *et al.*, "Application of decision tree technology for image classification using remote sensing data", *Agric. Syst.*, vol. 76, núm. 3, pp. 1101–1117, jun. 2003, doi: 10.1016/S0308-521X(02)00051-3.
[43] R. SHARMA, A. GHOSH, y P. K. JOSHI, "Decision tree approach for classification of remotely sensed satellite data using open source support", *J. Earth Syst. Sci.*, vol. 122, núm. 5, pp. 1237–1247, oct. 2013, doi: 10.1007/s12040-013-0339-2.
[44] M. A. Friedl y C. E. Brodley, "Decision tree classification of land cover from remotely sensed data", *Remote Sens. Environ.*, vol. 61, núm. 3, pp. 399–409, sep. 1997, doi: 10.1016/S0034-4257(97)00049-7.
[45] M. Mehmood, A. Shahzad, B. Zafar, A. Shabbir, y N. Ali, "Remote Sensing Image Classification: A Comprehensive Review and Applications", *Math. Probl. Eng.*, vol. 2022, p. e5880959, ago. 2022, doi: 10.1155/2022/5880959.
[46] F. Roli y G. Fumera, "Support vector machines for remote sensing image classification", en *Image and Signal Processing for Remote Sensing VI*, SPIE, ene. 2001, pp. 160–166. doi: 10.1117/12.413892.
[47] G. Mountrakis, J. Im, y C. Ogole, "Support vector machines in remote sensing: A review", *ISPRS J. Photogramm. Remote Sens.*, vol. 66, núm. 3, pp. 247–259, may 2011, doi: 10.1016/j.isprsjprs.2010.11.001.
[48] F. Melgani y L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines", *IEEE Trans. Geosci. Remote Sens.*, vol. 42, núm. 8, pp. 1778–1790, ago. 2004, doi: 10.1109/TGRS.2004.831865.
[49] P. Du, J. Xia, W. Zhang, K. Tan, Y. Liu, y S. Liu, "Multiple Classifier System for Remote Sensing Image Classification: A Review", *Sensors*, vol. 12, núm. 4, Art. núm. 4, abr. 2012, doi: 10.3390/s120404764.

[50] M. Han, X. Zhu, y W. Yao, “Remote sensing image classification based on neural network ensemble algorithm”, *Neurocomputing*, vol. 78, núm. 1, pp. 133–138, feb. 2012, doi: 10.1016/j.neucom.2011.04.044.
[51] G. P. Joshi, F. Alenezi, G. Thirumoorthy, A. K. Dutta, y J. You, “Ensemble of Deep Learning-Based Multimodal Remote Sensing Image Classification Model on Unmanned Aerial Vehicle Networks”, *Mathematics*, vol. 9, núm. 22, Art. núm. 22, ene. 2021, doi: 10.3390/math9222984.
[52] A. Krizhevsky, I. Sutskever, y G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, *Commun. ACM*, vol. 60, núm. 6, pp. 84–90, dic. 2012, doi: 10.1145/3065386.
[53] K. Simonyan y A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *ArXiv14091556 Cs*, abr. 2015, Consultado: el 22 de marzo de 2022. [En línea]. Disponible en: http://arxiv.org/abs/1409.1556
[54] E. Maggiori, Y. Tarabalka, G. Charpiat, y P. Alliez, “Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification”, *IEEE Trans. Geosci. Remote Sens.*, vol. 55, núm. 2, pp. 645–657, feb. 2017, doi: 10.1109/TGRS.2016.2612821.
[55] W. Rawat y Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review”, *Neural Comput.*, vol. 29, núm. 9, pp. 2352–2449, sep. 2017, doi: 10.1162/neco_a_00990.
[56] X. Sun *et al.*, “FAIR1M: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery”, *ISPRS J. Photogramm. Remote Sens.*, vol. 184, pp. 116–130, feb. 2022, doi: 10.1016/j.isprsjprs.2021.12.004.
[57] E. Maggiori, Y. Tarabalka, G. Charpiat, y P. Alliez, “Fully convolutional neural networks for remote sensing image classification”, en *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Beijing, China: IEEE, jul. 2016, pp. 5071–5074. doi: 10.1109/IGARSS.2016.7730322.
[58] Y. Lecun, L. Bottou, Y. Bengio, y P. Haffner, “Gradient-based learning applied to document recognition”, *Proc. IEEE*, vol. 86, núm. 11, pp. 2278–2324, nov. 1998, doi: 10.1109/5.726791.
[59] M. Castelluccio, G. Poggi, C. Sansone, y L. Verdoliva, “Land Use Classification in Remote Sensing Images by Convolutional Neural Networks”. arXiv, el 1 de agosto de 2015. doi: 10.48550/arXiv.1508.00092.
[60] D. Matthew Zeiler y F. Rob, “Visualizing and understanding convolutional neural networks”, ECCV, 2014.
[61] N. Dalal y B. Triggs, “Histograms of Oriented Gradients for Human Detection”, presentado en International Conference on Computer Vision & Pattern Recognition (CVPR ’05), IEEE Computer Society, jun. 2005, p. 886. doi: 10.1109/CVPR.2005.177.

[62] P. A. Torrione, K. D. Morton, R. Sakaguchi, y L. M. Collins, “Histograms of Oriented Gradients for Landmine Detection in Ground-Penetrating Radar Data”, *IEEE Trans. Geosci. Remote Sens.*, vol. 52, núm. 3, pp. 1539–1550, mar. 2014, doi: 10.1109/TGRS.2013.2252016.
[63] Vincent Mbandu Ochango, Geoffrey Mariga Wambugu, y John Gichuki Ndia, “Feature Extraction using Histogram of Oriented Gradients for Image Classification in Maize Leaf Diseases”, *Int. J. Comput. Inf. Technol.-0764*, vol. 11, núm. 2, jun. 2022, doi: 10.24203/ijcit.v11i2.204.
[64] T. Ojala, M. Pietikainen, y T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, núm. 7, pp. 971–987, jul. 2002, doi: 10.1109/TPAMI.2002.1017623.
[65] L. Huang, C. Chen, W. Li, y Q. Du, “Remote Sensing Image Scene Classification Using Multi-Scale Completed Local Binary Patterns and Fisher Vectors”, *Remote Sens.*, vol. 8, núm. 6, Art. núm. 6, jun. 2016, doi: 10.3390/rs8060483.
[66] M. Wang *et al.*, “Assessing Texture Features to Classify Coastal Wetland Vegetation from High Spatial Resolution Imagery Using Completed Local Binary Patterns (CLBP)”, *Remote Sens.*, vol. 10, núm. 5, Art. núm. 5, may 2018, doi: 10.3390/rs10050778.
[67] A. Farooq, X. Jia, J. Hu, y J. Zhou, “Multi-Resolution Weed Classification via Convolutional Neural Network and Superpixel Based Local Binary Pattern Using Remote Sensing Images”, *Remote Sens.*, vol. 11, núm. 14, Art. núm. 14, ene. 2019, doi: 10.3390/rs11141692.
[68] D. G. Lowe, “Object recognition from local scale-invariant features”, en *Proceedings of the Seventh IEEE International Conference on Computer Vision*, sep. 1999, pp. 1150–1157 vol.2. doi: 10.1109/ICCV.1999.790410.
[69] Q. Li, G. Wang, J. Liu, y S. Chen, “Robust Scale-Invariant Feature Matching for Remote Sensing Image Registration”, *IEEE Geosci. Remote Sens. Lett.*, vol. 6, núm. 2, pp. 287–291, abr. 2009, doi: 10.1109/LGRS.2008.2011751.
[70] H. Deng, L. Wang, J. Liu, D. Li, Z. Chen, y Q. Zhou, “Study on Application of Scale Invariant Feature Transform Algorithm on Automated Geometric Correction of Remote Sensing Images”, en *Computer and Computing Technologies in Agriculture VI*, D. Li y Y. Chen, Eds., en IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer, 2013, pp. 352–358. doi: 10.1007/978-3-642-36137-1_41.
[71] Y. LeCun *et al.*, “Handwritten Digit Recognition with a Back-Propagation Network”, en *Advances in Neural Information Processing Systems*, Morgan-Kaufmann, 1989. Consultado: el 6 de agosto de 2023. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html

[72] M. Mahdianpari, B. Salehi, M. Rezaee, F. Mohammadimanesh, y Y. Zhang, “Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery”, *Remote Sens.*, vol. 10, núm. 7, Art. núm. 7, jul. 2018, doi: 10.3390/rs10071119.
[73] D. Yu, Q. Xu, H. Guo, C. Zhao, Y. Lin, y D. Li, “An Efficient and Lightweight Convolutional Neural Network for Remote Sensing Image Scene Classification”, *Sensors*, vol. 20, núm. 7, Art. núm. 7, ene. 2020, doi: 10.3390/s20071999.
[74] S. Bera y V. K. Shrivastava, “Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification”, *Int. J. Remote Sens.*, vol. 41, núm. 7, pp. 2664–2683, abr. 2020, doi: 10.1080/01431161.2019.1694725.
[75] Y. Long, Y. Gong, Z. Xiao, y Q. Liu, “Accurate Object Localization in Remote Sensing Images Based on Convolutional Neural Networks”, *IEEE Trans. Geosci. Remote Sens.*, vol. 55, núm. 5, pp. 2486–2498, may 2017, doi: 10.1109/TGRS.2016.2645610.
[76] R. Alshehhi, P. R. Marpu, W. L. Woon, y M. D. Mura, “Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks”, *ISPRS J. Photogramm. Remote Sens.*, vol. 130, pp. 139–149, ago. 2017, doi: 10.1016/j.isprsjprs.2017.05.002.
[77] Q. Zhang, Q. Yuan, C. Zeng, X. Li, y Y. Wei, “Missing Data Reconstruction in Remote Sensing Image With a Unified Spatial–Temporal–Spectral Deep Convolutional Neural Network”, *IEEE Trans. Geosci. Remote Sens.*, vol. 56, núm. 8, pp. 4274–4288, ago. 2018, doi: 10.1109/TGRS.2018.2810208.
[78] Z. Shao, P. Tang, Z. Wang, N. Saleem, S. Yam, y C. Sommai, “BRRNet: A Fully Convolutional Neural Network for Automatic Building Extraction From High-Resolution Remote Sensing Images”, *Remote Sens.*, vol. 12, núm. 6, Art. núm. 6, ene. 2020, doi: 10.3390/rs12061050.
[79] M. Rezaee, M. Mahdianpari, Y. Zhang, y B. Salehi, “Deep Convolutional Neural Network for Complex Wetland Classification Using Optical Remote Sensing Imagery”, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, núm. 9, pp. 3030–3039, sep. 2018, doi: 10.1109/JSTARS.2018.2846178.
[80] S. Ji, C. Zhang, A. Xu, Y. Shi, y Y. Duan, “3D Convolutional Neural Networks for Crop Classification with Multi-Temporal Remote Sensing Images”, *Remote Sens.*, vol. 10, núm. 1, Art. núm. 1, ene. 2018, doi: 10.3390/rs10010075.
[81] D. A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*. John Wiley & Sons, 2003.
[82] G.-S. Xia *et al.*, “AID: A Benchmark Dataset for Performance Evaluation of Aerial Scene Classification”, *IEEE Trans. Geosci. Remote Sens.*, vol. 55, núm. 7, pp. 3965–3981, jul. 2017, doi: 10.1109/TGRS.2017.2685945.

[83] W. Zhou, S. Newsam, C. Li, y Z. Shao, “PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval”, *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, nov. 2018, doi: 10.1016/j.isprsjprs.2018.01.004.
[84] H. Li *et al.*, “RSI-CB: A Large Scale Remote Sensing Image Classification Benchmark via Crowdsource Data”. arXiv, el 10 de enero de 2020. doi: 10.48550/arXiv.1705.10450.
[85] Y. Di, Z. Jiang, y H. Zhang, “A Public Dataset for Fine-Grained Ship Classification in Optical Remote Sensing Images”, *Remote Sens.*, vol. 13, núm. 4, Art. núm. 4, ene. 2021, doi: 10.3390/rs13040747.
[86] A. E. Maxwell, T. A. Warner, y F. Fang, “Implementation of machine-learning classification in remote sensing: an applied review”, *Int. J. Remote Sens.*, vol. 39, núm. 9, pp. 2784–2817, may 2018, doi: 10.1080/01431161.2018.1433343.
[87] X. Zhang, H. Liangxiu, H. Lianghao, y Z. Liang, “How Well Do Deep Learning-Based Methods for Land Cover Classification and Object Detection Perform on High Resolution Remote Sensing Imagery?”, ene. 2020, Consultado: el 19 de junio de 2023. [En línea]. Disponible en: https://www.mdpi.com/2072-4292/12/3/417
[88] M. Wieland y M. Pittore, “Performance Evaluation of Machine Learning Algorithms for Urban Pattern Recognition from Multi-spectral Satellite Images”, *Remote Sens.*, vol. 6, núm. 4, Art. núm. 4, abr. 2014, doi: 10.3390/rs6042912.
[89] G. Ge, Z. Shi, Y. Zhu, X. Yang, y Y. Hao, “Land use/cover classification in an arid desert-oasis mosaic landscape of China using remote sensed imagery: Performance assessment of four machine learning algorithms”, *Glob. Ecol. Conserv.*, vol. 22, p. e00971, jun. 2020, doi: 10.1016/j.gecco.2020.e00971.
[90] R. Khatami, G. Mountrakis, y S. V. Stehman, “A meta-analysis of remote sensing research on supervised pixel-based land-cover image classification processes: General guidelines for practitioners and future research”, *Remote Sens. Environ.*, vol. 177, pp. 89–100, may 2016, doi: 10.1016/j.rse.2016.02.028.
[91] Z. Shao, K. Yang, y W. Zhou, “Performance Evaluation of Single-Label and Multi-Label Remote Sensing Image Retrieval Using a Dense Labeling Dataset”, *Remote Sens.*, vol. 10, núm. 6, Art. núm. 6, jun. 2018, doi: 10.3390/rs10060964.
[92] H. Li, S. Zhang, X. Ding, C. Zhang, y P. Dale, “Performance Evaluation of Cluster Validity Indices (CVIs) on Multi/Hyperspectral Remote Sensing Datasets”, *Remote Sens.*, vol. 8, núm. 4, Art. núm. 4, abr. 2016, doi: 10.3390/rs8040295.
[93] M. Ahmad *et al.*, “Hyperspectral Image Classification—Traditional to Deep Models: A Survey for Future Prospects”, *IEEE J. Sel. Top. Appl. Earth Obs.*

Remote Sens., vol. 15, pp. 968–999, 2022, doi: 10.1109/JSTARS.2021.3133021.
[94] A. K. Jain y A. Vailaya, “Image retrieval using color and shape”, *Pattern Recognit.*, vol. 29, núm. 8, pp. 1233–1244, ago. 1996, doi: 10.1016/0031-3203(95)00160-3.
[95] A. Oliva y A. Torralba, “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope”, *Int. J. Comput. Vis.*, vol. 42, núm. 3, pp. 145–175, may 2001, doi: 10.1023/A:1011139631724.
[96] W. S. McCulloch y W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bull. Math. Biophys.*, vol. 5, núm. 4, pp. 115–133, dic. 1943, doi: 10.1007/BF02478259.
[97] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain”, *Psychol. Rev.*, vol. 65, núm. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.
[98] Rosenblat, “PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS”, 1962. Consultado: el 6 de agosto de 2023. [En línea]. Disponible en: https://apps.dtic.mil/sti/citations/AD0256582
[99] S. Skansi, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. en Undergraduate Topics in Computer Science. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-73004-2.
[100] A. K. Jain, J. Mao, y K. M. Mohiuddin, “Artificial neural networks: a tutorial”, *Computer*, vol. 29, núm. 3, pp. 31–44, mar. 1996, doi: 10.1109/2.485891.
[101] Y. LeCun, “A Theoretical Framework for Back-Propagation”, en *Proceedings of the 1988 connectionist models summer school*, San Mateo, CA, USA: Touresky, D and Hinton, G and Sejnowski, T, 1988, pp. 21–28. Consultado: el 10 de agosto de 2023. [En línea]. Disponible en: https://www.researchgate.net/profile/Yann-Lecun/publication/2360531_A_Theoretical_Framework_for_Back-Propagation/links/0deec519dfa297eac1000000/A-Theoretical-Framework-for-Back-Propagation.pdf
[102] D. E. Rumelhart, G. E. Hinton, y R. J. Williams, “Learning representations by back-propagating errors”, *Nature*, vol. 323, núm. 6088, Art. núm. 6088, oct. 1986, doi: 10.1038/323533a0.
[103] Alaeldin Suliman y Yun Zhang, “A Review on Back-Propagation Neural Networks in the Application of Remote Sensing Image Classification”, *J. Earth Sci. Eng.*, vol. 5, núm. 1, ene. 2015, doi: 10.17265/2159-581X/2015.01.004.
[104] I. Goodfellow, Y. Bengio, y A. Courville, *Deep Learning*. MIT Press, 2016.
[105] G. E. Hinton y R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks”, *Science*, vol. 313, núm. 5786, pp. 504–507, jul. 2006, doi: 10.1126/science.1127647.

[106] G. Cheng, X. Xie, J. Han, L. Guo, y G.-S. Xia, “Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities”, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 3735–3756, 2020, doi: 10.1109/JSTARS.2020.3005403.
[107] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-94463-0.
[108] A. Fischer y C. Igel, “An Introduction to Restricted Boltzmann Machines”, en *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, L. Alvarez, M. Mejail, L. Gomez, y J. Jacobo, Eds., en Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 14–36. doi: 10.1007/978-3-642-33275-3_2.
[109] G. E. Hinton, S. Osindero, y Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets”, *Neural Comput.*, vol. 18, núm. 7, pp. 1527–1554, jul. 2006, doi: 10.1162/neco.2006.18.7.1527.
[110] B. Ghojogh, A. Ghodsi, F. Karray, y M. Crowley, “Restricted Boltzmann Machine and Deep Belief Network: Tutorial and Survey”. arXiv, el 5 de agosto de 2022. doi: 10.48550/arXiv.2107.12521.
[111] Y. LeCun, Y. Bengio, y G. Hinton, “Deep learning”, *Nature*, vol. 521, núm. 7553, Art. núm. 7553, may 2015, doi: 10.1038/nature14539.
[112] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, y F. E. Alsaadi, “A survey of deep neural network architectures and their applications”, *Neurocomputing*, vol. 234, pp. 11–26, abr. 2017, doi: 10.1016/j.neucom.2016.12.038.
[113] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, y R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, *JMLR Org*, vol. 15, núm. 1, pp. 1929--1958, 2014.
[114] C. Szegedy *et al.*, “Going Deeper With Convolutions”, presentado en Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9. Consultado: el 2 de diciembre de 2022. [En línea]. Disponible en: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html
[115] M. Lin, Q. Chen, y S. Yan, “Network In Network”. arXiv, el 4 de marzo de 2014. doi: 10.48550/arXiv.1312.4400.
[116] K. He, X. Zhang, S. Ren, y J. Sun, “Deep Residual Learning for Image Recognition”, arXiv, arXiv:1512.03385, dic. 2015. doi: 10.48550/arXiv.1512.03385.
[117] Z. Wu, C. Shen, y A. van den Hengel, “Wider or Deeper: Revisiting the ResNet Model for Visual Recognition”, *Pattern Recognit.*, vol. 90, pp. 119–133, jun. 2019, doi: 10.1016/j.patcog.2019.01.006.
[118] M. Kubat, *An Introduction to Machine Learning*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-63913-0.

[119] M. Grandini, E. Bagli, y G. Visani, “Metrics for Multi-Class Classification: an Overview”. arXiv, el 13 de agosto de 2020. doi: 10.48550/arXiv.2008.05756.
[120] S. S. Nath, G. Mishra, J. Kar, S. Chakraborty, y N. Dey, “A survey of image classification methods and techniques”, en *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, jul. 2014, pp. 554–557. doi: 10.1109/ICCICCT.2014.6993023.
[121] B. T. Chicho y A. B. Sallow, “A Comprehensive Survey of Deep Learning Models Based on Keras Framework”, *J. Soft Comput. Data Min.*, vol. 2, núm. 2, Art. núm. 2, oct. 2021.
[122] “Google Colaboratory”. Consultado: el 20 de octubre de 2022. [En línea]. Disponible en: https://colab.research.google.com/
[123] B. Qu, X. Li, D. Tao, y X. Lu, “Deep semantic understanding of high resolution remote sensing image”, en *2016 International Conference on Computer, Information and Telecommunication Systems (CITS)*, jul. 2016, pp. 1–5. doi: 10.1109/CITS.2016.7546397.
[124] Z. Reitermanová, “Reitermanova: Data splitting - Google Scholar”. Consultado: el 1 de noviembre de 2022. [En línea]. Disponible en: https://scholar.google.com/scholar_lookup?hl=en&publication_year=2010&pages=31-36&author=Z.+Reitermanov%C3%A1&title=%E2%80%9CData+Splitting
[125] “Data augmentation | TensorFlow Core”, TensorFlow. Consultado: el 23 de octubre de 2023. [En línea]. Disponible en: https://www.tensorflow.org/tutorials/images/data_augmentation
[126] “Keras: Deep Learning for humans”. Consultado: el 23 de octubre de 2023. [En línea]. Disponible en: https://keras.io/

ANEXOS

ANEXO I

Modelo VGG-16 & conjunto UCM

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_max_pooling2d (GlobalMaxPooling2D)	(None, 512)	0
dense (Dense)	(None, 4096)	2101248
batch_normalization (BatchNormalization)	(None, 4096)	16384
dense_1 (Dense)	(None, 4096)	16781312
batch_normalization_1 (BatchNormalization)	(None, 4096)	16384
dense_2 (Dense)	(None, 21)	86037

Total params: 33,716,053

Trainable params: 18,984,981

Non-trainable params: 14,731,072

Modelo VGG-16 & conjunto Sydney

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_max_pooling2d (GlobalMaxPooling2D)	(None, 512)	0
dense (Dense)	(None, 4096)	2101248
batch_normalization (BatchNormalization)	(None, 4096)	16384
dense_1 (Dense)	(None, 4096)	16781312
batch_normalization_1 (BatchNormalization)	(None, 4096)	16384
dense_2 (Dense)	(None, 7)	28679

Total params: 33,658,695

Trainable params: 18,927,623

Non-trainable params: 14,731,072

Modelo VGG-16 & conjunto RSICD

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_max_pooling2d (GlobalMaxPooling2D)	(None, 512)	0
batch_normalization (BatchNormalization)	(None, 512)	2048
dense (Dense)	(None, 4096)	2101248
batch_normalization_1 (BatchNormalization)	(None, 4096)	16384
dense_1 (Dense)	(None, 4096)	16781312
batch_normalization_2 (BatchNormalization)	(None, 4096)	16384
dense_2 (Dense)	(None, 30)	122910

Total params: 33,754,974

Trainable params: 19,022,878

Non-trainable params: 14,732,096

Modelo ResNet-50 & UCM

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
batch_normalization (BatchNormalization)	(None, 7, 7, 2048)	8192
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
batch_normalization_1 (BatchNormalization)	(None, 1024)	4096
dense_1 (Dense)	(None, 21)	21525

Total params: 25,719,701

Trainable params: 2,125,845

Non-trainable params: 23,593,856

Modelo ResNet-50 & Sydney

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
batch_normalization (BatchNormalization)	(None, 7, 7, 2048)	8192
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
batch_normalization_1 (BatchNormalization)	(None, 1024)	4096
dense_1 (Dense)	(None, 7)	7175

Total params: 25,705,351

Trainable params: 2,111,495

Non-trainable params: 23,593,856

Modelo ResNet-50 & RSICD

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
batch_normalization (BatchNormalization)	(None, 7, 7, 2048)	8192
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
batch_normalization_1 (BatchNormalization)	(None, 2048)	8192
dense (Dense)	(None, 1024)	2098176
batch_normalization_2 (BatchNormalization)	(None, 1024)	4096
dense_1 (Dense)	(None, 30)	30750

Total params: 25,737,118

Trainable params: 2,139,166

Non-trainable params: 23,597,952

Modelo Inception-V3 & UCM

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
dense_1 (Dense)	(None, 21)	21525

Total params: 23,922,485

Trainable params: 2,119,701

Non-trainable params: 21,802,784

Modelo Inception-V3 & Sydney

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
batch_normalization_94 (BatchNormalization)	(None, 1024)	4096
dense_1 (Dense)	(None, 7)	7175

Total params: 23,912,231

Trainable params: 2,107,399

Non-trainable params: 21,804,832

Modelo Inception-V3 & RSICD

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
batch_normalization_94 (BatchNormalization)	(None, 1024)	4096
dense_1 (Dense)	(None, 30)	30750

Total params: 23,935,806

Trainable params: 2,130,974

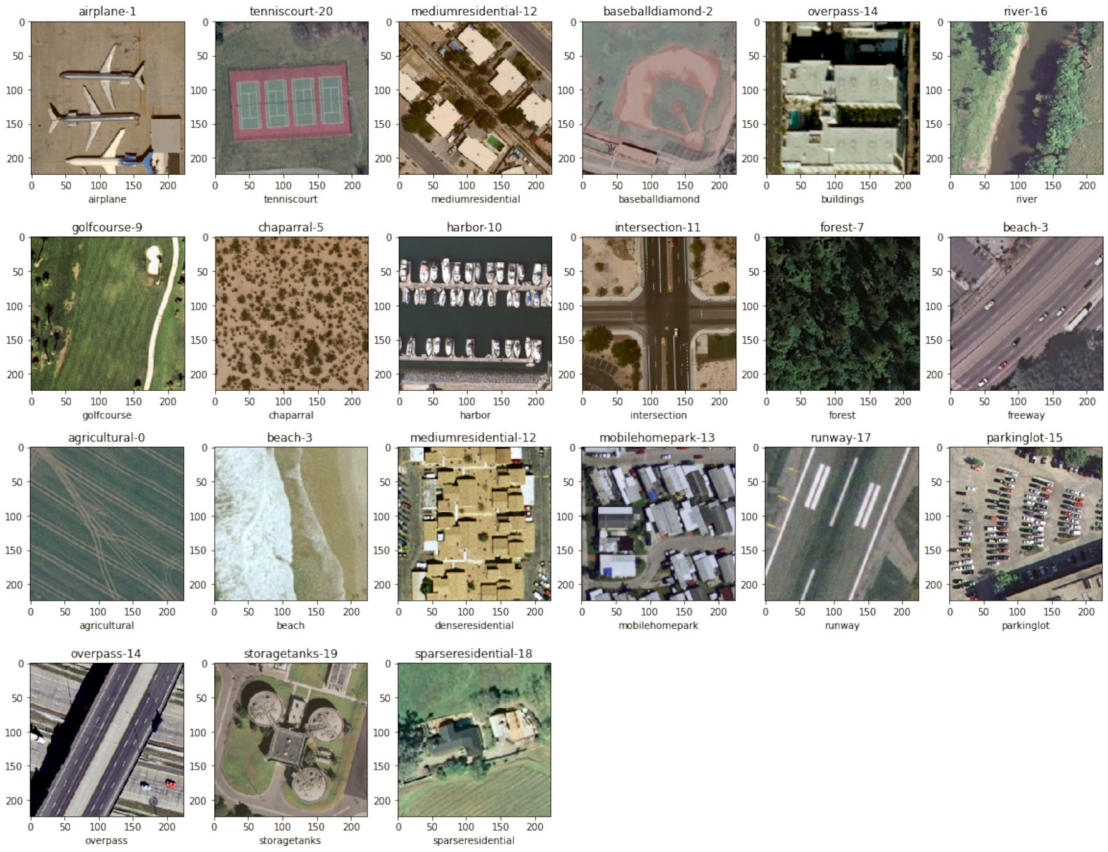
Non-trainable params: 21,804,832

ANEXO II

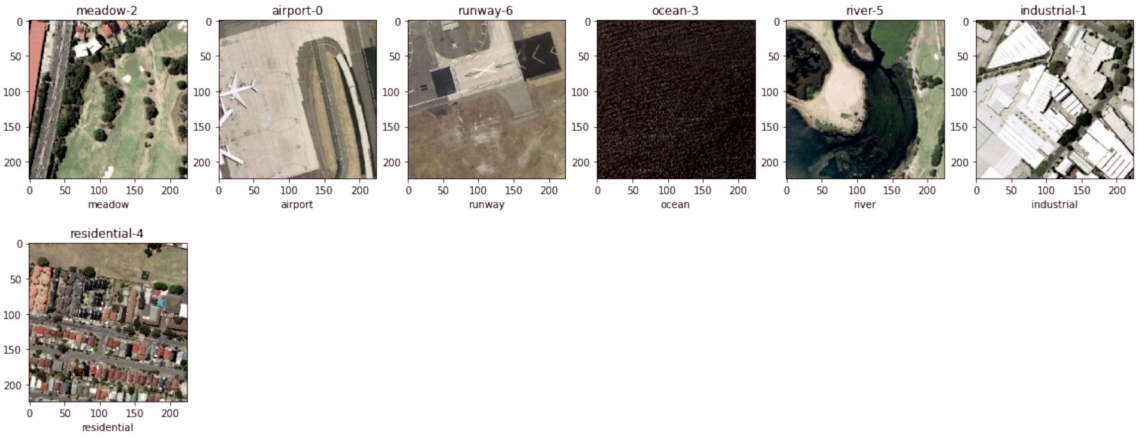
En cada una de las figuras del anexo 2, se visualiza una imagen de cada clase de cada conjunto de imágenes. En cada imagen individual, se observan dos etiquetas. La etiqueta en la parte inferior nos muestra el nombre verdadero de la clase a la que pertenece la imagen; la etiqueta de la parte superior, nos indica la predicción de la clase que ha entregado el algoritmo de predicción. Cuando las dos etiquetas entregan la misma clase, entonces la predicción del algoritmo es correcta y es falsa en el caso de que sean diferentes.

Procesamiento con GPU

Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación VGG-16, sobre el conjunto de imágenes UCM.

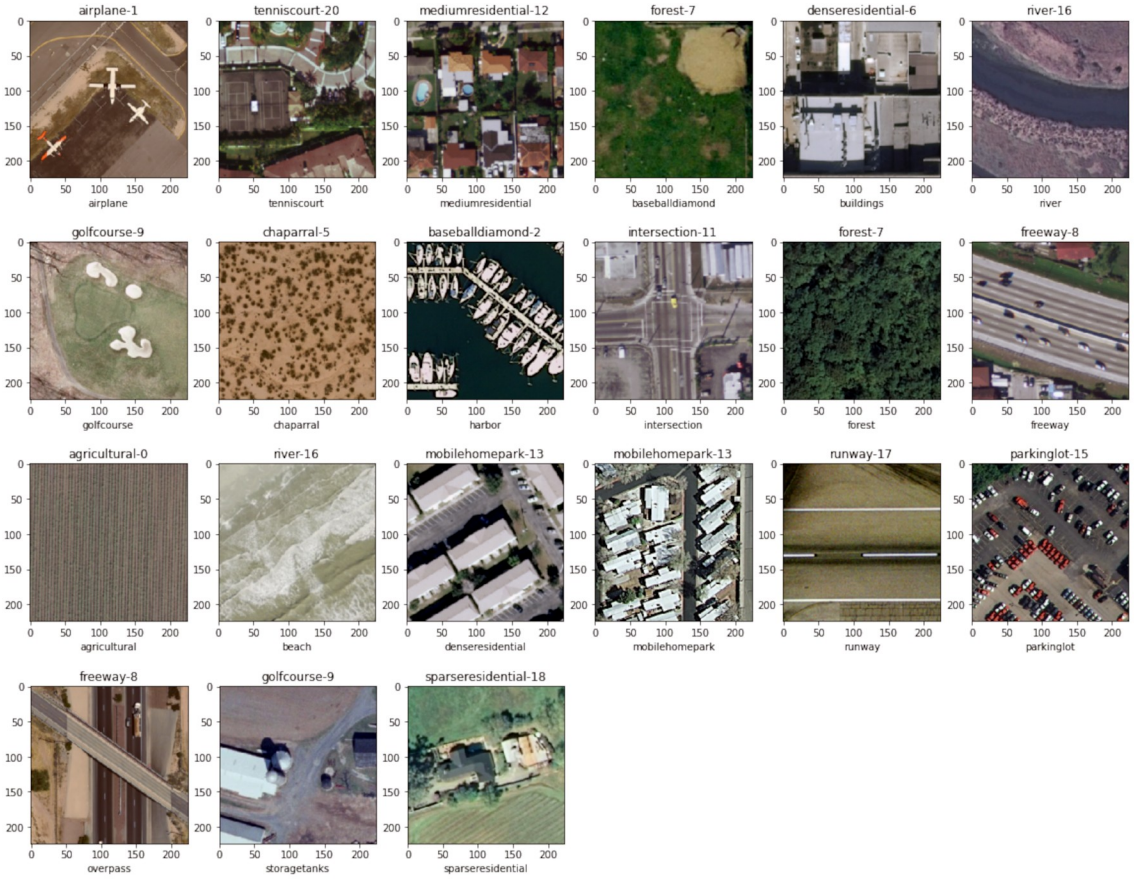


Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación VGG-16, sobre el conjunto de imágenes Sydney.

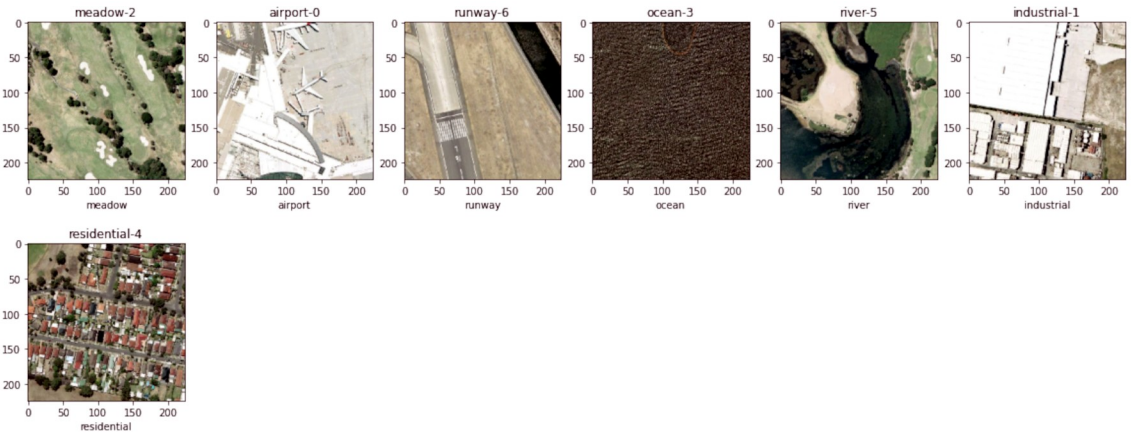


Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación VGG-16, sobre el conjunto de imágenes RSICD.

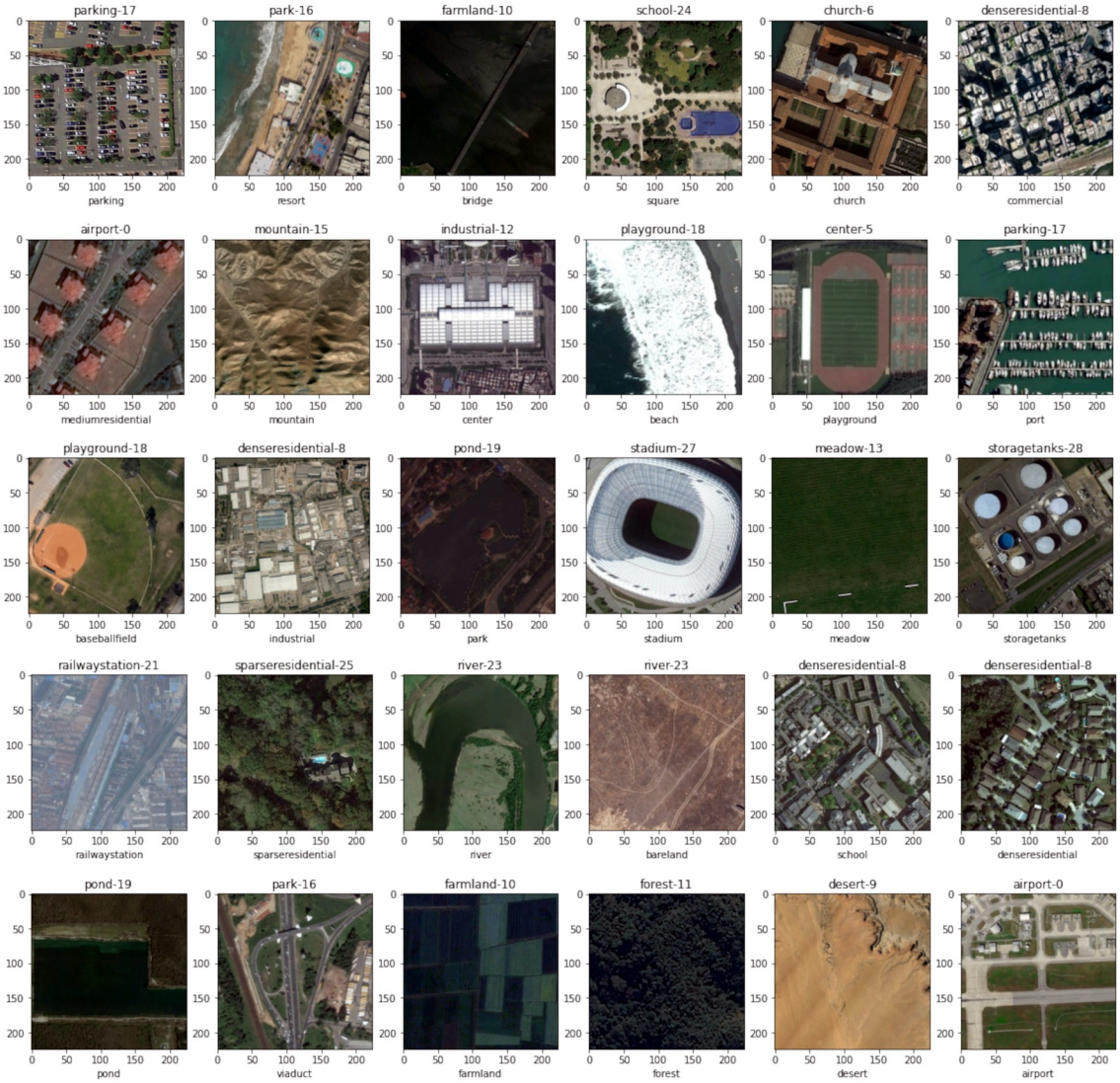
Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación ResNet-50, sobre el conjunto de imágenes UCM.



Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación ResNet-50, sobre el conjunto de imágenes Sydney.

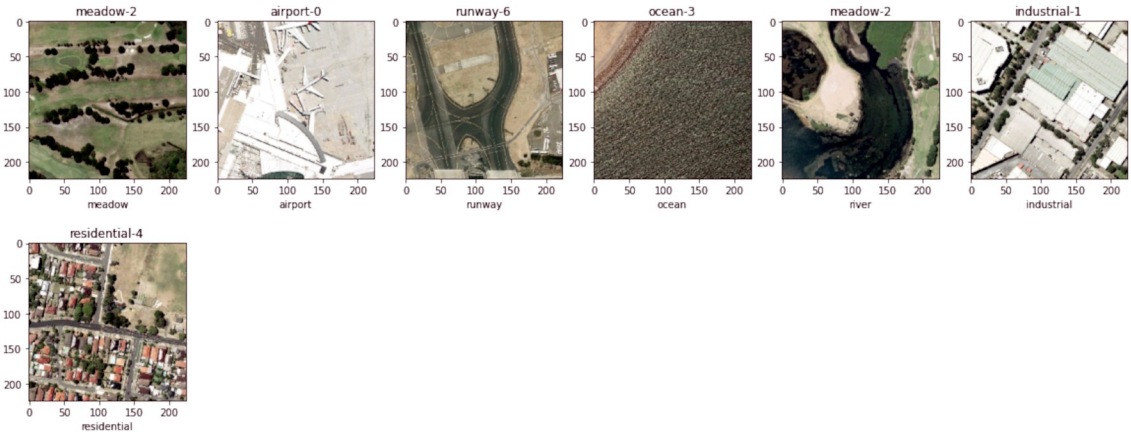


Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación ResNet-50, sobre el conjunto de imágenes RSICD.

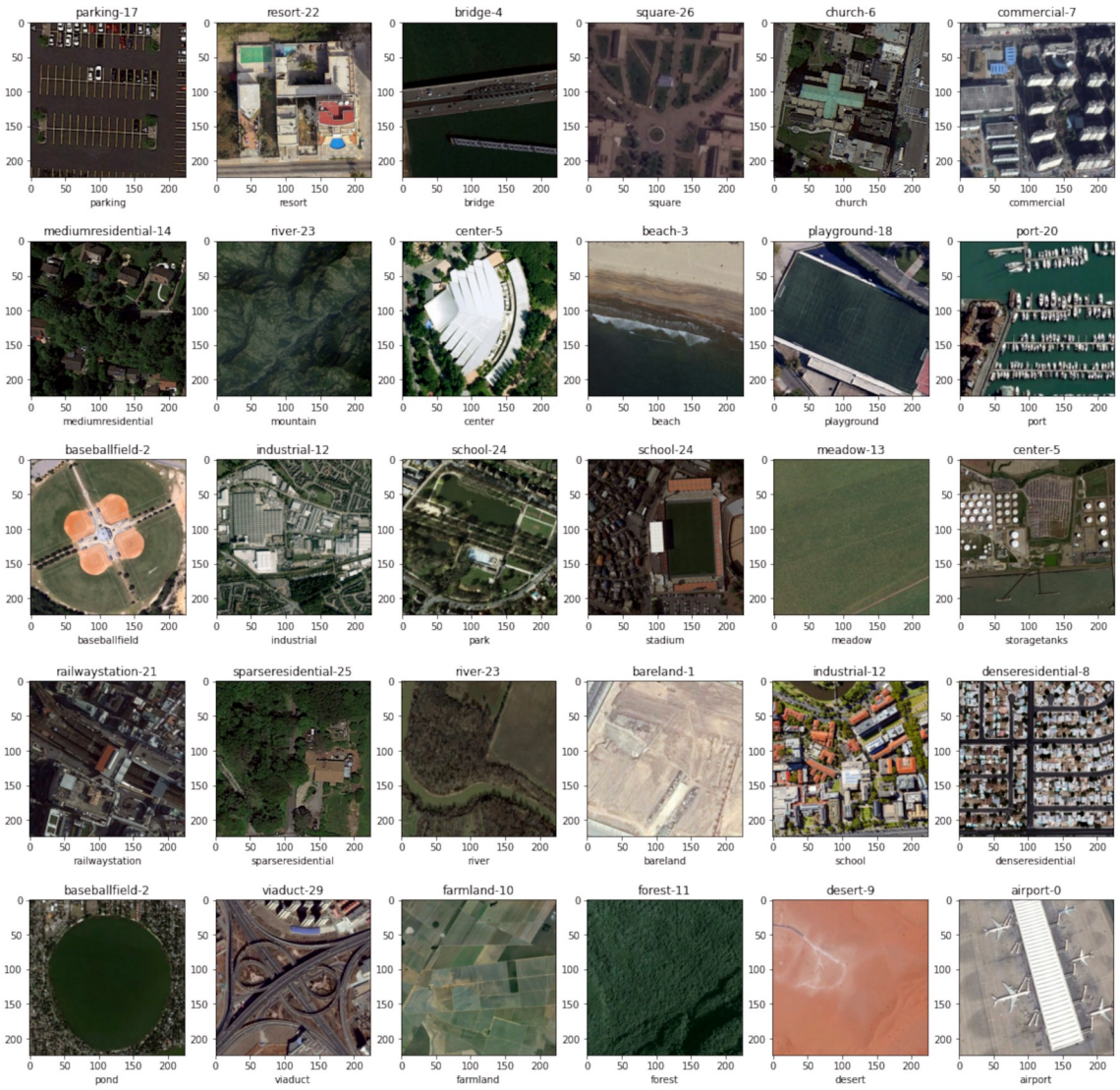


Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación Inception-v3, sobre el conjunto de imágenes UCM.

Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación Inception-v3, sobre el conjunto de imágenes Sydney.



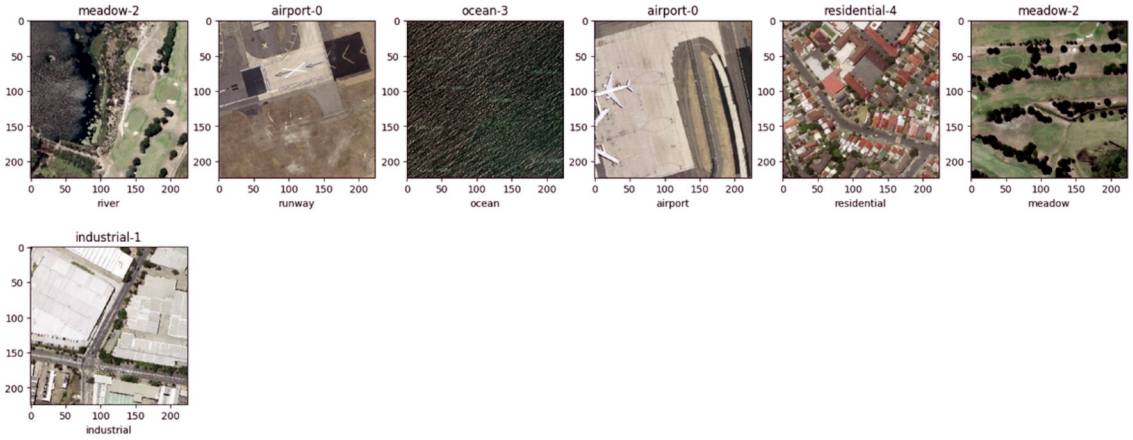
Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación Inception-v3, sobre el conjunto de imágenes RSICD.



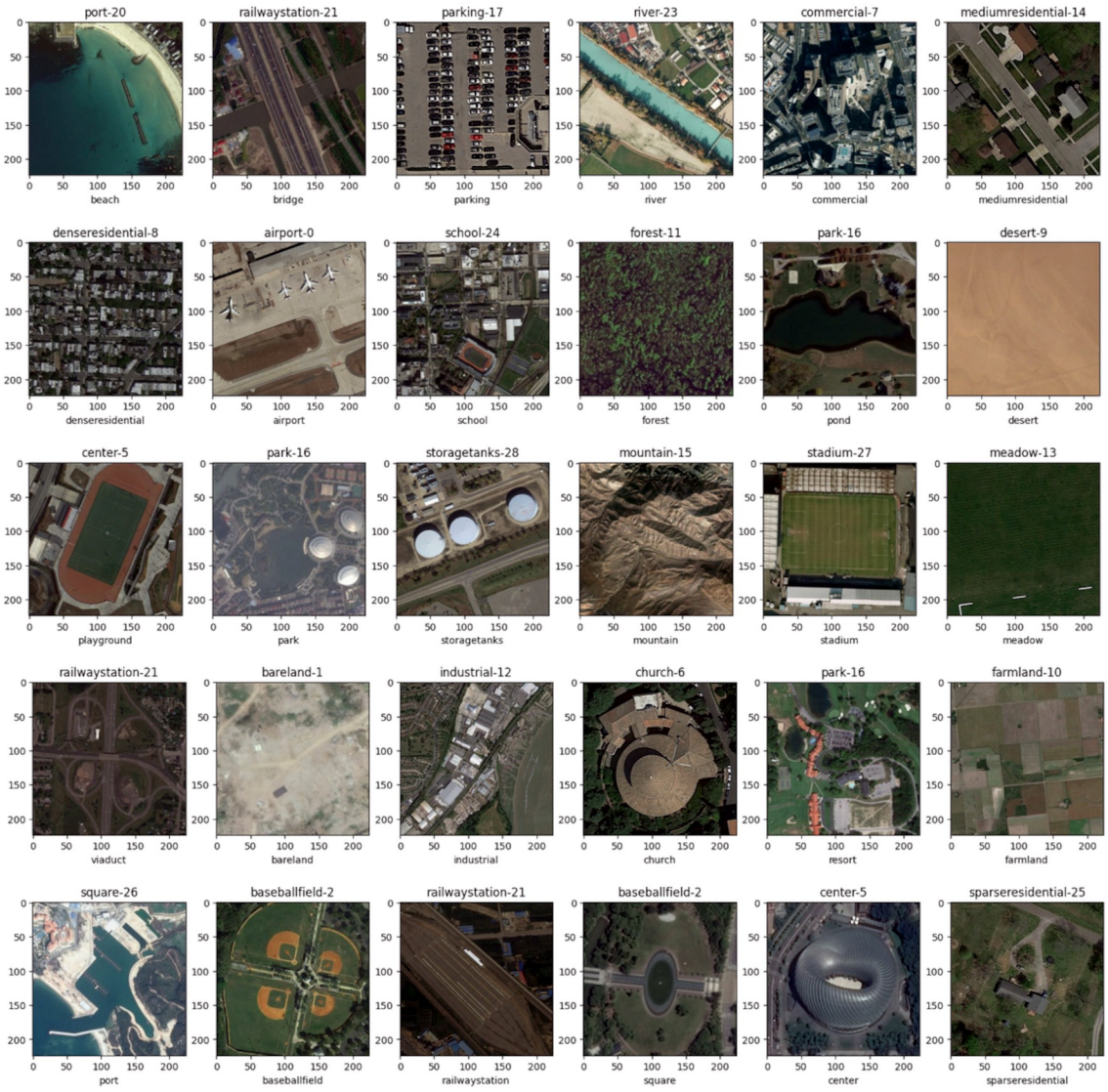
Procesamientos con CPU

Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación VGG-16, sobre el conjunto de imágenes UCM.

Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación VGG-16, sobre el conjunto de imágenes Sydney.

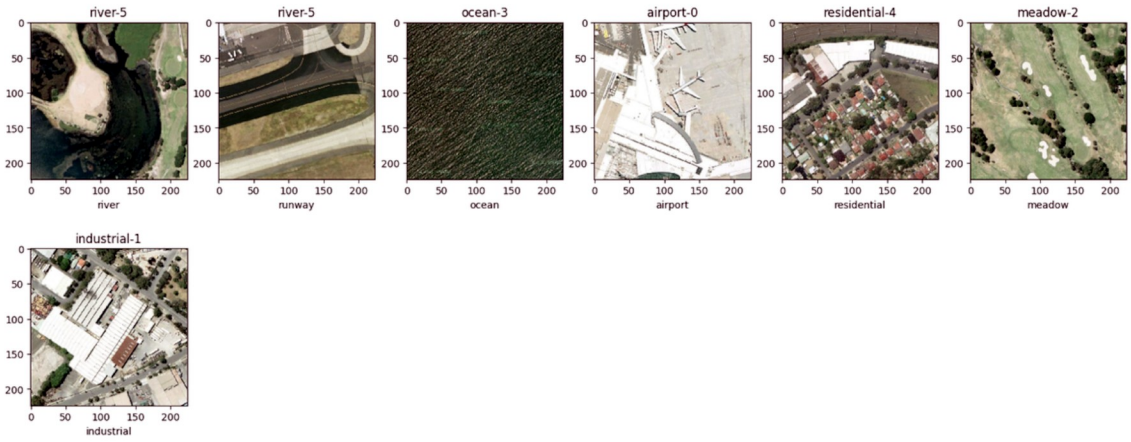


Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación VGG-16, sobre el conjunto de imágenes RSICD.

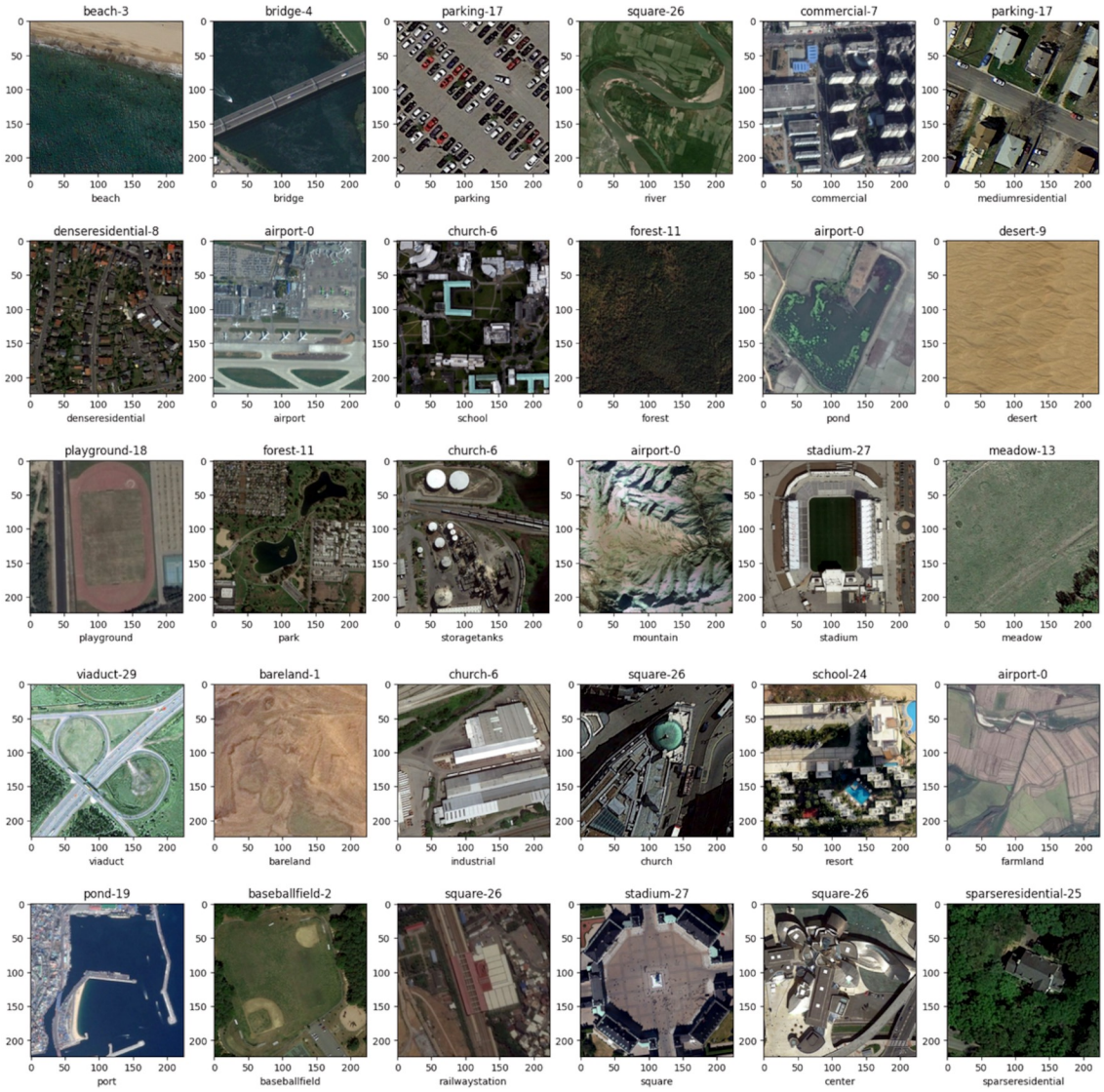


Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación ResNet-50, sobre el conjunto de imágenes UCM.

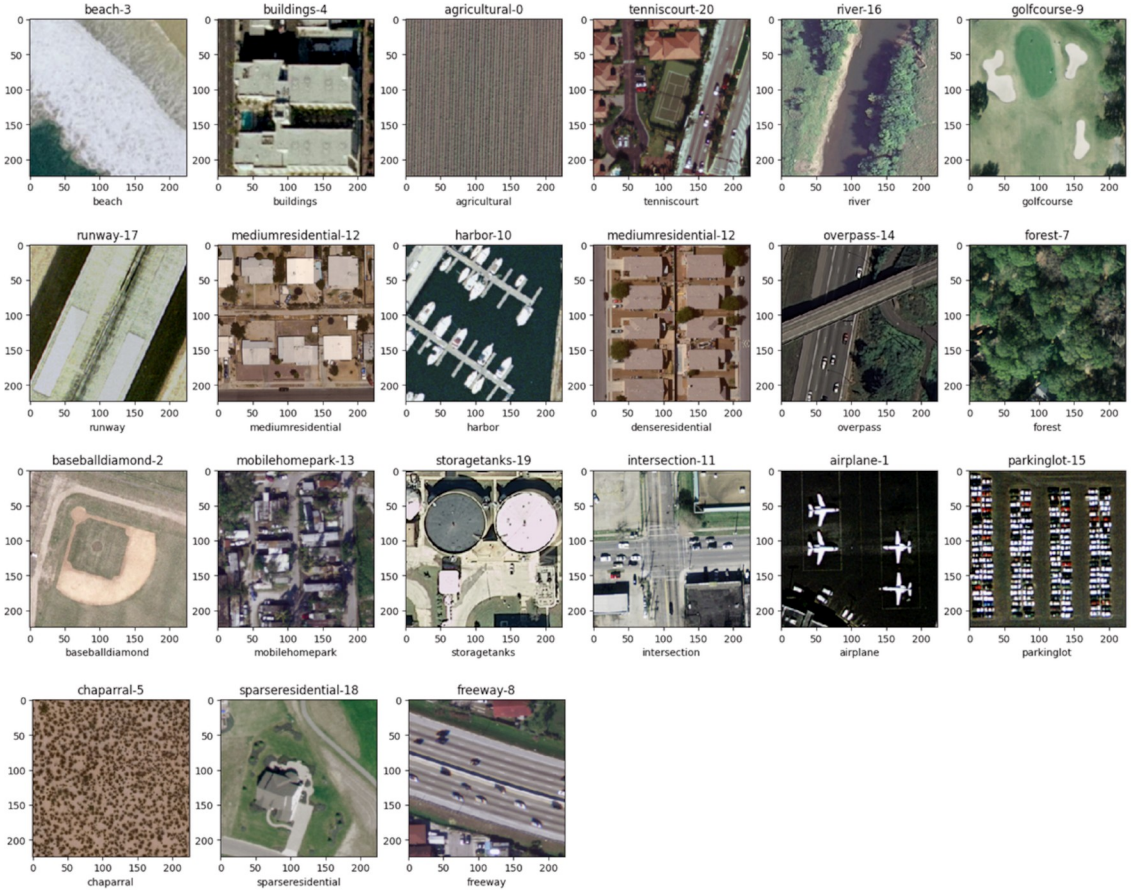
Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación ResNet-50, sobre el conjunto de imágenes Sydney.



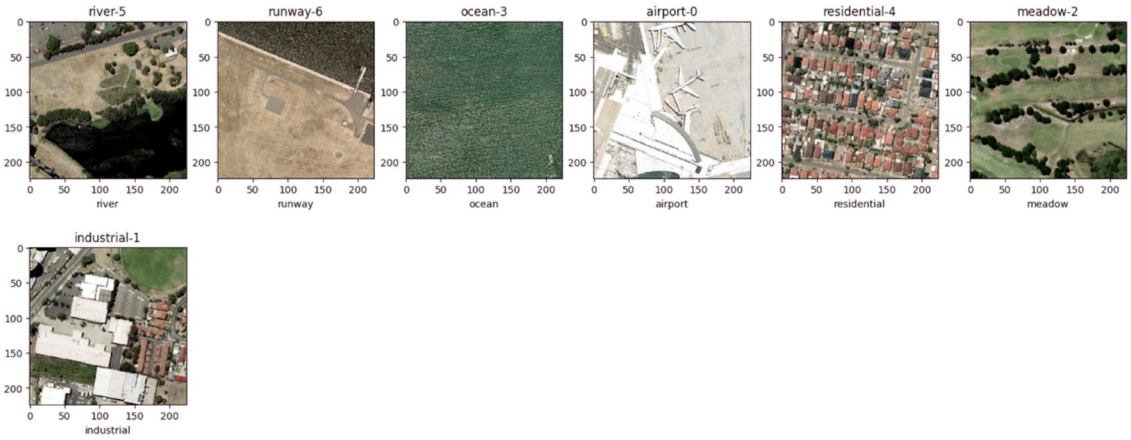
Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación ResNet-50, sobre el conjunto de imágenes RSICD.



Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación Inception-v3, sobre el conjunto de imágenes UCM.



Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación Inception-v3, sobre el conjunto de imágenes Sydney.



Ejemplo de visualización para la predicción de imágenes remotas con el modelo de clasificación Inception-v3, sobre el conjunto de imágenes RSICD.