DR. JEAN MICHEL GRÉVY MACQUART
COORDINADOR DEL POSGRADO EN CIENCIAS
PRESENTE

Atendiendo a la solicitud para emitir DICTAMEN sobre la revisión de la TESIS titulada: **Aprendizaje Profundo para la Clasificación de Imágenes de Plantas de Albahaca con Diferencias de Nitrógeno durante su Cultivo, Utilizando la Técnica de Entrenamiento Curriculum-By-Smoothing,** que presenta el alumno **Roger Salinas González (10036246)** para obtener el título de **Maestro en Ciencias.**

Nos permitimos informarle que nuestro voto es:

NOMBRE	DICTAMEN	FIRMA
Dra. Lorena Díaz González CInC-UAEM	APROBADO	
Dr. Jorge Hermosillo Valadez CInC-UAEM	APROBADO	
Dr. Edgar Francisco Román Rangel ITAM	APROBADO	
Dr. Porfirio Juárez López FCA-UAEM	APROBADO	
Dr. Jorge Alberto Fuentes Pacheco CInC-UAEM	APROBADO	

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

PORFIRIO JUAREZ LOPEZ | Fecha:2021-11-02 15:34:43 | Firmante

IkQSG191allV0WxHml/sv1Kx+nGa79TeVT7vYJpJOhlOFA9p64Chh/H1AXwsK1V6MALb0MDNQHSRpAWQAO+A3A13bSXOeVM7sr2zrCBVprveM5pOhgv9CuzqYphCtvKvOPma/Q+MzkCpjMFQjtYnGlKaY34HL4bwg5JZrq8GS0ae3Pa90rCA2lM7a/iljSC4f+aY1aJF1Pj5uyt6tR17B2zq0JGwPYlk30WJmI2/peCsNXdI7rYCPvyve3RxakpngBdA5XgxMx3jkLnYflS30u3V1wnHfLJtFWoBVCHJAt1gflTpi8taVhYAUR2Zn6xxbQ5gIQ7jyP+YcXwjUDZm+w==

JORGE HERMOSILLO VALADEZ | Fecha:2021-11-02 16:00:46 | Firmante

hVwlP+b2MS8FDlq9idWSEd5wc3BA7SgAjJh4VCiR8dkSwL2z5ggm4nbgdsJJHpctKLAH9DGdE90NzVxHxzO5pONx1TM0+ioV7fH5nCp0VkUnHF6v1+UKYu9A+d6f6JDXrd8GME+zVkYkQl0J6FvL4OBcj5Su1p2+s8XdmDDGsZVSZR1riwLGWLfIqVpkrwpgW20O2CosNyFbULrcv6re+FmEq9RRNlWkOKNVNV6EL8vjuTy4kWudsIbDVW4i4jNJDy7a+0MeV1C8cUrWkiM8R7JCwssCc0XTtfSh3Gx3WWi6tvf0oW65MbMKSpgpSK9G7jyY3LmBUsMbjEMiN3iKzw==

JORGE ALBERTO FUENTES PACHECO | Fecha:2021-11-11 21:37:58 | Firmante

UVp8AQLubjh1T1zT7grxIK3ov67rU5yFx8FcdkzX+x9e6Z6rDkO+vM7FesfTx+sxUx+sMHmSnHICAZ5BuQMGhiiDfk44OPT+4beCjNAqGT5U63H6oPEUPVHtH7VuNl5iNC83Q/8UBhG/8o5JF/sP1qNA10Fz1Xa8mZ7sO1X/cKZdv/4uIsahGYu320UCMuF9xNy6c4KHW1Kla/Ps3dAWRgbotNMmBtfjq6nvsaBVaJhuS+dH9oDcV/SeLWFRxd+TNcHM28vy24Vlhg+k82Ww5RB8ksg2wLC2jqY+ChIoc72unhVUPhldpRvt7jl7H5mDm+wThbdDgmDCELsNNZPfIA==

LORENA DIAZ GONZALEZ | Fecha:2021-11-16 13:21:36 | Firmante

Te6bQPp2aCX5GnUIgDOMxRcreL9stFJTEdFFaEdgDBwhn/Gu1Lv39pFVy3rS7DU37Ruxg7IkGoC4eIfSvQzd0oWbWdRbtCW6N08/MwgAt33do4N4RKYE9DhKJPcG47vQCLEqMgqgp0WlrP3hSFclZwxmSS2garwvevJEWS+nBhHI+04dHo4QAtiiq7SKBO8z8rAZfZA7fHaR57PpE8RJOcc7v2P47HpSiBnlG2IGKXIMZp6HwlNWuiaMz0DLfbLzoRwHQNBPRc5m62aY54+OxRV7yvYswi5aOIBOEV47VFFGRxwcI+SsIRQUyxdI6khqDTZd20K+fTK8FHd1K04uXA==

EDGAR FRANCISCO ROMÁN RANGEL | Fecha:2021-11-18 14:30:03 | Firmante

X/Xdh9XFxkU2cPWHQzA3wi8UEte9lnAfiFacVmH1juqOHXs85OcoxPO1Y40Re4m4kffNYi89y0AlNSDsjXqnrX8yNRKZZNtDt4ooylvxYEK1s6QuQejVrbsJqxy2qP1A6mHZSV0XDMXl1yDfbzccQNtZcIEXE7CUDCu1HkdyO8fQWZvI+ElznTsupg75XawQq/dme4KuhVf086qw4L8H364THQH4NX+IBRJ2JeIcXcf/kFdjba8gXygLe/ADFNseJQt7Mq/G167+II4tPQlK/FJAx4MBpoU5DLvoCvotBxcNNQ56ryz7/D3AmkSfzQd5rSNvMTQlma2/dRjrZrzG/A==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

DVQ5BIJUH

https://efirma.uaem.mx/noRepudio/RRguB3NzfXYdUBVF7o4HVgUvthOtlCzR

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS

"APRENDIZAJE PROFUNDO PARA LA CLASIFICACIÓN DE IMÁGENES DE PLANTAS DE ALBAHACA CON DIFERENCIAS DE NITRÓGENO DURANTE SU CULTIVO, UTILIZANDO LA TÉCNICA DE ENTRENAMIENTO CURRICULUM-BY-SMOOTHING"

Tesis

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS

PRESENTA

ROGER SALINAS GONZÁLEZ

DIRECTOR DE TESIS

Dr. Juan Manuel Rendón Mancha

CODIRECTOR DE TESIS

Dr. Jorge Alberto Fuentes Pacheco

CUERNAVACA, MORELOS SEPTIEMBRE, 2021

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los últimos años hemos asistido a una verdadera revolución en el campo de la inteligencia artificial. Estos avances, producidos en la última década, se deben en gran medida a las técnicas basadas en Aprendizaje Profundo (Deep Learning, DL) con Redes Neuronales Convolucionales (CNN) y a la mejora del hardware que hace posible su uso a gran escala. Así, las Redes Neuronales han demostrado ser la mejor opción conocida, hasta ahora, de una gran variedad de aplicaciones de la inteligencia artificial. Estos algoritmos tienen como característica su capacidad de aprender de forma autónoma, realizan un aprendizaje supervisado en el que durante su entrenamiento obtienen una función adaptada a unos datos previamente etiquetados. Las redes neuronales, por tanto, son capaces de aprender con base a una información suministrada previamente, generalmente imágenes previamente clasificadas cuyas características son aprendidas por la red, sin que el desarrollador tenga que intervenir en cómo lo hace. El DL ha mostrado un éxito sobresaliente en casi todos los dominios de aplicaciones. Se pueden citar como ejemplo: Localización de objetos, Descripción de imágenes o vídeos, Juegos, Segmentación de imagen o vídeo, Conducción autónoma de vehículos, Traducción de textos, Reconocimiento de voz, Seguridad y defensa, Medicina y biología. En este contexto las técnicas basadas en DL con Redes Neuronales Convolucionales CNN, muestran la precisión más avanzada en la tarea de reconocimiento visual usando la base de datos ImageNet [Alom y col., 2019].

Recientemente se ha propuesto un nuevo esquema de entrenamiento o método de aprendizaje, llamado Currículo por Suavizado (Curriculum-by-Smoothing, CbS), que propone controlar y exponer progresivamente la información relacionada con la textura de las imágenes, durante el proceso de entrenamiento. Este método de entrenamiento crea un currículo para que la red aprenda primero las características de la forma de los objetos presentes en las imágenes con que se entrena y gradualmente le va aumentado la cantidad de información de textura que le es permitida analizar a la red. La red neuronal está compuesta por neuronas de entrada y a cada una se le atribuye un peso pero si a partir de los mismos valores de entrada queremos que el valor de la predicción sea diferente y que se active la neurona, es necesario adicionar una neurona que tiene su propio peso y que de alguna manera va a forzar la predicción para algunos valores de entrada, esto es lo que se denomina un sesgo. Las redes con un mayor sesgo de forma son intrínsecamente más robustas para muchas distorsiones de imagen diferentes y alcanzan un mayor rendimiento en tareas de clasificación y reconocimiento de

objetos [Geirhos y col., 2018].

El uso de CNN en la agricultura puede encontrarse en disímiles tareas como: detección de plagas, de enfermedades en las plantas, predicción de suelos y clasificación de cultivos. El método de estudio tradicional para afrontar estos problemas generalmente pasa por llevar muestras de tejido vegetativo a un laboratorio especializado, mover a un ingeniero agrónomo experto al sitio del cultivo o realizar análisis químicos empleando equipos costosos. Sin embargo, el uso redes neuronales convolucionales es cada vez más tendencioso para este tipo de problemáticas, pues se obtienen diagnósticos cercanos a lo que un experto humano determinaría [Ortega y col., 2019]. El único problema sería que el entrenamiento de la red es demasiado costoso computacionalmente y se requiere una gran cantidad de datos para hacerlo. En el contexto de esta investigación demostramos que la técnica entrenamiento o método de aprendizaje llamado currículo por suavizado mejora, considerablemente, los resultados de clasificación que modelos de redes neuronales del estado del arte, como ResNet50[V2] y VGG16, obtienen en la tarea de clasificar imágenes de plantas de albahaca identificando cuatro clases posibles resultado de la aplicación de diferentes concentraciones de Nitrógeno durante el cultivo controlado de la planta. Se puede constatar que se alcanzan elevados valores de exactitud, en la tarea de clasificación, aún considerando el reducido número de imágenes de que se dispone.

1.1. Problema de la investigación

Necesidad de obtener una buena clasificación de imágenes de plantas de albahaca cultivadas con diferentes concentraciones de nitrógeno, utilizando una base de datos de 144 imágenes de entrenamiento y 72 imágenes de validación por cada clase evaluada utilizando modelos de Redes Neuronales de gran capacidad del estado del arte, como son VGG16 y ResNet50V2, con inicialización de pesos en la base de datos ImageNet.

1.2. Hipótesis

Fijar el entrenamiento del modelo de red bajo el enfoque curriculum-by-smoothing garantiza mejores valores de los parámetros que caracterizan el aprendizaje, logrando una mejor clasificación en función de las clases a las que pertenece cada una de las imágenes de plantas de albahaca.

1.3. Objetivo general de la investigación

Utilizar la técnica de entrenamiento curriculum-by-smoothing en los modelos de redes neuronales VGG16 y ResNet50[V2] para analizar el desempeño en la clasificación de las imágenes, en comparación con la técnica convencional de transferencia de aprendizaje, en una base de datos de imágenes de plantas de albahaca que fueron tratadas con diferentes concentraciones de Nitrógeno durante su cultivo.

1.4. Objetivos específicos

Como objetivos específicos planteados para esta tesis se encuentran:

1. Implementar la técnica de currículo por suavizado sobre modelos clásicos de redes neuronales del estado del arte para la clasificación de plantas de albahaca.

2. Cuantificar si este esquema de entrenamiento mejora significativamente el rendimiento del modelo de red, sin agregar ningún parámetro entrenable, en la tarea de la clasificación de las imágenes de plantas de albahaca.

1.5. Organización del documento

Después de esta introducción que constituye el capítulo 1 el documento consta de cinco capítulos más:

• el capítulo 2 ofrece una breve redacción de los antecedentes e historia de las redes neuronales para pasar después a describir su arquitectura, entrenamiento supervisado, características de las principales capas que la componen, detalles de los dos modelos de estudio y las métricas y métodos para la visualización e interpretación de un modelo de red entrenado.

• el capítulo 3 introduce los conceptos de aprendizaje por transferencia, aprendizaje curricular y plan de estudios por textura.

• en el capitulo 4 la implementación del método de aprendizaje para los modelos de estudio y la evaluación del modelo a través de la matriz de confusión y de las imágenes de localización basada en gradientes.

• el capítulo 5 presenta los resultados obtenidos detallando las características de la base de datos usada y se visualizan las gráficas para los diferentes entrenamientos.

• por último en el capítulo 6 las conclusiones del trabajo.

Capítulo 2

Fundamento teórico

2.1. Antecedentes

2.1.1. Redes Neuronales

Las redes neuronales artificiales son un modelo computacional que trata de imitar el funcionamiento del cerebro humano y aunque no se conoce exactamente cómo éste funciona, sí que se saben ciertas características como la estructura que tiene, e inspirándose en esas observaciones surgen los algoritmos que la desarrollan. Estas redes, como es de esperar, funcionan de una manera abstracta y simple y hoy en día, dentro del ámbito del aprendizaje de máquinas (Machine Learning), se han convertido en una de las técnicas más conocidas y con más renombre. Concretamente el algoritmo consiste en un conjunto de unidades denominadas neuronas conectadas entre sí para transmitirse información de unas a otras. Cuando los datos/información de entrada atraviesan esta red de neuronas, se acaba produciendo un resultado o unos valores de salida que son los que nos interesan.

De este modo se puede entender una red neuronal como una gran función que produce una salida dependiente de unos datos de entrada, pero dicha función para ello se apoya en multitud de sub-funciones que la componen. Es decir, se puede ver una red neuronal como una función de funciones que permiten a la función principal computar y capturar potentes relaciones en los datos. Las sub-funciones son transformaciones no lineales, la función completa es una cascada de transformaciones no lineales. El objetivo final es que dicha gran función acabe realizando predicciones correctas para los datos de entrada, es decir que la salida obtenida por la red sea la esperada. Las redes son métodos supervisados para problemas de regresión multivariada, y se consideran funciones de aproximación universal.

Su funcionamiento se podría simplificar de la siguiente forma: Al recibir una serie de entradas numéricas, se realizan un conjunto de cálculos internos que terminan dando como resultado otros datos numéricos que pueden estar representados en otro espacio. En el caso de las imágenes a partir de una entrada numérica (las imágenes ya son datos numéricos internamente), obtener una salida que podría ser la probabilidad de que en dicha imagen se reconozca la existencia de una característica determinada.

2.1.2. Historia de las Redes Neuronales

Sus orígenes se remontan a 1943, año en el que W. McCulloch y W. Pitts crearon un primer modelo de redes neuronales basándose en las matemáticas y algoritmos denominados lógica de umbral [McCulloch y W., 1943]. Poco después de sus avances, Donald Hebb explicó a grandes rasgos el aprendizaje neuronal, conociéndose su ley como la "Regla de Hebb ", [Hebb, 1949] la cual es precursora de las técnicas de aprendizaje de las redes neuronales en la actualidad. Posteriormente, en 1956, se produjo en Dartmouth la primera conferencia en la que se habló de la capacidad de la computación de simular el aprendizaje de los cerebros biológicos. Al poco de esto (1960) llegó la primera aplicación de las redes a problemas reales: La eliminación de ecos en líneas telefónicas mediante el Adaline (Adaptative Linear Elements) y el Madaline (Multiple Adaptative Linear Elements).

Sin embargo, en 1969, con la publicación del libro "Perceptrons: An introduction to Computational Geometry" [Minsky y Papert, 1969] el interés por las redes neuronales cayó repentinamente dado que en el libro se demostraron importantes deficiencias en los modelos neuronales artificiales que se habían desarrollado hasta ese momento, sobre todo del perceptrón. A pesar de esto la investigación no se frenó y en 1977 se destacó el trabajo realizado por James Anderson "Brain-State-in-a-Box" [Anderson y col., 1977], el cual permitió modelar funciones de mayor complejidad. El resurgimiento de las redes neuronales se produjo en 1985 con la publicación por parte de John Hopfield del libro "Computación neuronal de decisiones en problemas de optimización" [Hopfield y Tank, 1985]; un año más tarde, en [D. Rumelhart y Williams, 1986] los autores introdujeron en su versión actual el algoritmo de propagación hacia atrás (backpropagation), lo cual provocó un nuevo panorama mucho más alentador en las investigaciones y desarrollo de redes neuronales.

Desde entonces la investigación en el campo de las Redes Neuronales ha experimentado un crecimiento vertiginoso, produciéndose avances tanto en software como en hardware. Gracias al algoritmo de propagación hacia atrás se hizo posible entrenar redes neuronales de múltiples capas de manera supervisada siendo así que en 1989 aparecen las denominadas redes convolucionales -CNN Convolutional Neural Network- redes de varias capas que toman su inspiración en la corteza visual de los animales. La primera CNN fue creada por Yann LeCun y estaba enfocada en el reconocimiento de letras manuscritas [LeCun y col., 1989]. Esta arquitectura demostró ser útil en varias aplicaciones, principalmente en procesamiento de imágenes. La arquitectura constaba de varias capas que implementaban la extracción de características y luego la clasificación. Esta arquitectura usando capas profundas y la clasificación de salida abrieron un mundo nuevo de posibilidades en las redes neuronales.

En 1997 se crearon las LSTM -Long Short Term Memory- un tipo de red neuronal recurrente, una arquitectura que permite conexiones "hacia atrás" entre las capas, las que consisten en unas celdas de memoria que permiten a la red recordar valores por períodos cortos o largos [Schmidhuber y Hochreiter, 1997].

Una celda de memoria contiene compuertas que administran cómo la información fluye dentro o fuera. La puerta de entrada controla cuándo puede entrar nueva información en la memoria. La puerta de "olvido" controla cuanto tiempo existe y se retiene esa información. La puerta de salida controla cuando la información en la celda es usada como salida de la celda. La celda contiene pesos que controlan cada compuerta. El algoritmo de entrenamiento

conocido como backpropagation-through-time optimiza estos pesos basado en el error del resultado. Las LSTM se han aplicado en reconocimiento de voz, de escritura, text-to-speech y otras tareas.

A pesar de los avances citados los modelos con "profundidad" (decenas o cientos de capas) eran considerados demasiado difíciles de entrenar (incluso con backpropagation) y el uso de las redes neuronales artificiales quedó estancado hasta la creación en el 2006 de las redes de creencias profundas (Deep Belief Networks, DBN) [Hochreiter y Schmidhuber, 1997], un modelo gráfico generativo, o alternativamente un tipo de red neuronal profunda, compuesta por múltiples capas de variables latentes, con conexiones entre las capas pero no entre unidades dentro de cada capa. Las redes DBN aprenden una capa a la vez al tratar los valores de las variables latentes en una capa, como los datos para entrenar la siguiente capa. Este aprendizaje ávido y eficiente puede ser seguido o combinado con otros procedimientos de aprendizaje que ajustan todos los pesos para mejorar el desempeño generativo o discriminativo de toda la red [Hinton, 2009]. Aunque al día de hoy las DBN no se utilizan mucho, fueron un gran hito en la historia para el desarrollo del deep learning y permitieron seguir la exploración para mejorar las redes existentes.

2.2. Componentes de una neurona artificial

Simplificando la representación de una neurona podemos decir que no es más que una serie de entradas, un conjunto de pesos y una función de activación. Su modelo se representa en la (Fig. (2.1)). El papel de la neurona es el de transformar dicha serie de entradas en un único valor de salida, el cual puede ser tomado como entrada de otras neuronas en capas superiores.

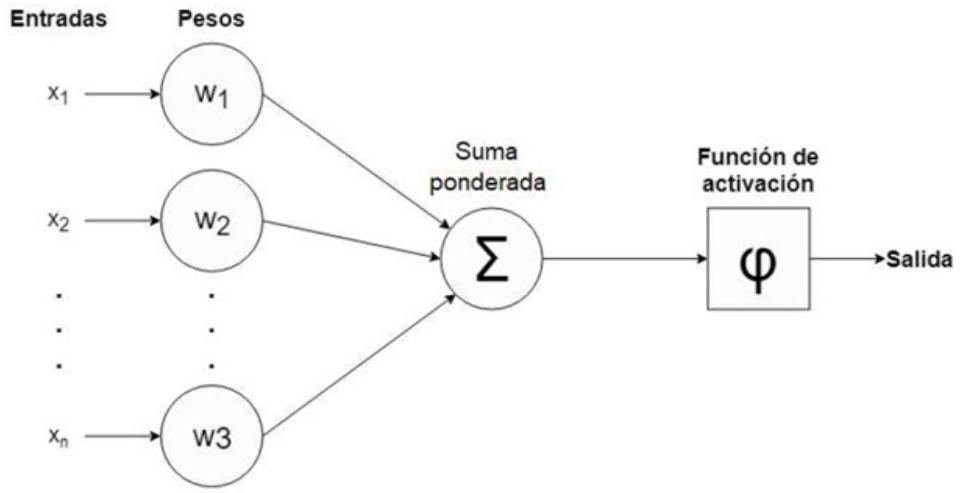


Figura 2.1: Modelo básico de una neurona artificial.

• Entradas: Pueden ser los datos numéricos iniciales o las salidas (también numéricas) de otras neuronas.

• Pesos: El factor de ponderación por el que se multiplicará la entrada correspondiente, estos pesos son propios de la neurona.

• Salida: El valor numérico que la neurona produce tras realizar diferentes cálculos.

• Suma ponderada: Cada entrada es multiplicada por su peso correspondiente, y se realiza el sumatorio de todas estas multiplicaciones. El valor resultante es la entrada a la función de activación. Al resultado del sumatorio se añade en ocasiones un bias. Que es, esencialmente, una nueva entrada con valor 1 y peso b.

• Función de activación: Su principal cometido es aportar no linealidad al modelo.

2.3. Función de activación

La función de activación define la salida de la neurona. Generalmente, se considera determinista y en la gran mayoría de los modelos es monótona creciente y continua. La elección de la misma es fundamental para fijar el control sobre el proceso de entrenamiento del modelo de red. También se suele referirse a la función de activación como una función de transferencia o llamarse función de aplastamiento cuando el rango de salida tiene limitaciones. Muchas (aunque no todas) de las transformaciones no lineales usadas en redes neuronales transforman los datos a un rango conveniente, por ejemplo [0,1] ó [-1,1].

Un ejemplo excluyente y muy usado de estos rangos de valores se obtiene de la función de activación lineal rectificada (Rectified Linear Unit, ReLU). Esta función de activación en su versión más simple se define por la ecuación(2.1) y su gráfico se puede apreciar en la (Figura. (2.2)) donde podemos apreciar que ReLU es el argumento positivo de la función y = x. En este caso el rango de valores es [0,∞].

$$\mathbf{f}(\mathbf{x}) = \text{máx}(0, x). \tag{2.1}$$

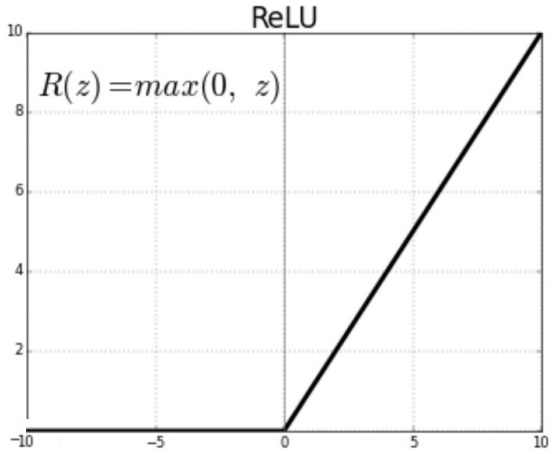


Figura 2.2: Función de activación lineal rectificada.

Otra de las formas de las funciones de activación más empleadas se pueden ver en la (Fig. (2.3)) [Haykin, 2009].

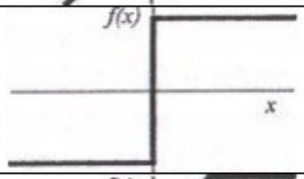
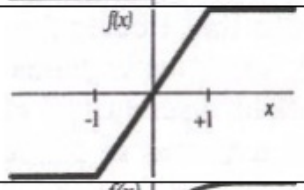
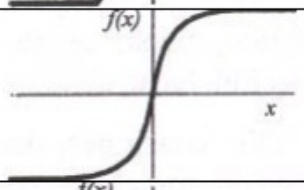
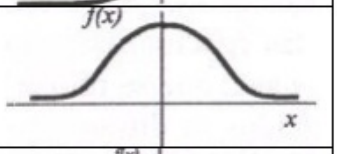
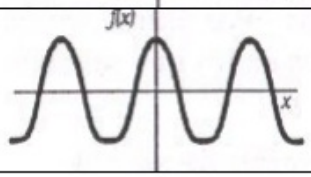
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, \infty]$	
Escalón	$y = \begin{cases} 1, & si\ x \geq 0 \\ 0, & si\ x < 0 \end{cases}$ $y = \begin{cases} 1, & si\ x \geq 0 \\ -1, & si\ x < 0 \end{cases}$	$[0, 1]$ $[-1, 1]$	
Lineal a tramos	$y = \begin{cases} 1, & si\ x > 1 \\ x, & si - 1 \leq x \leq 1 \\ -1, & x < -1 \end{cases}$	$[-1, 1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \tanh(x)$	$[0, 1]$ $[-1, 1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, 1]$	
Sinusoidal	$y = A\sin(wx + \varphi)$	$[-1, 1]$	

Figura 2.3: Funciones de activación.

En el desarrollo de diferentes tipos de redes neuronales, la función de activación lineal rectificada se está convirtiendo en la elección por defecto de los científicos de datos [Data-Science-Team, 2021].

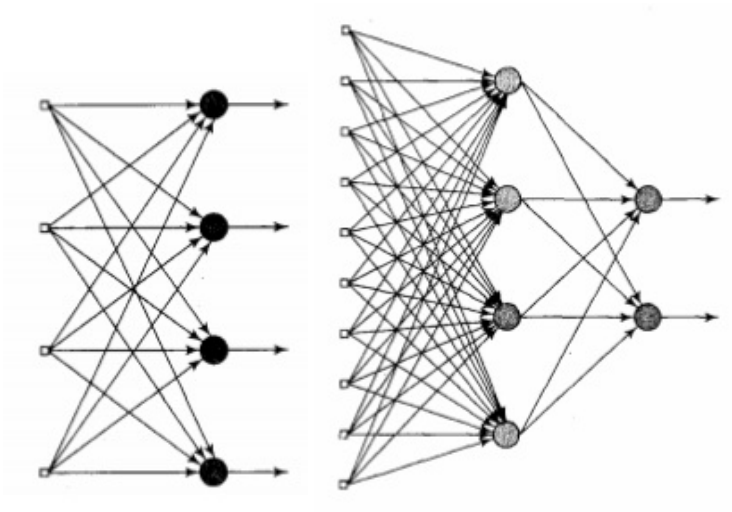
2.4. Arquitectura de las redes neuronales

2.4.1. Según el número de capas

En el diseño de una red neuronal se distinguen tres tipos de capas: de entrada, de salida y ocultas. La capa de entrada está compuesta por neuronas que reciben datos procedentes del entorno. La capa de salida se compone de neuronas que proporcionan la respuesta de la red neuronal. Las capas ocultas no tiene una conexión directa con el entorno. Este tipo de capa oculta proporciona grados de libertad a la red neuronal gracias a los cuales es capaz

de representar más fehacientemente determinadas características del entorno que trata de modelar.

Así considerando su estructura podemos hablar de redes monocapa –compuestas por una única capa de neuronas– o redes multicapa –las neuronas se organizan en varias capas: de entrada, de salida y ocultas (Fig. (2.4)) [Haykin, 2009].



(a) Red neuronal monocapa **(b)** Red neuronal multicapa

Figura 2.4: Clasificación según el número de capas.

2.4.2. Según la realimentación

Las redes neuronales según su realimentación pueden clasificarse en: [Haykin, 2009]

• Redes neuronales no recurrentes: en este tipo de redes, la propagación de las señales se produce en un único sentido.

• Redes neuronales recurrentes: de manera contraria a las anteriores, estas redes se caracterizan por poseer lazos de realimentación. Dichos lazos, pueden ser entre neuronas de diferentes capas, de la misma capa o incluso de una neurona consigo misma (Fig.(2.5)).

Una red neuronal recurrente no tiene una estructura de capas definida, sino que permite conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos, con esto se consigue crear la temporalidad, permitiendo que la red tenga memoria. Se puede entender esta temporalidad de forma sencilla apreciando que una capa de neuronas recurrentes se puede implementar de tal manera que, en cada instante de tiempo, cada neurona recibe dos entradas, la entrada correspondiente de la capa anterior y a su vez la salida del instante anterior de la misma capa (Fig.(2.6)).

Dado que la salida de una neurona recurrente en un instante de tiempo determinado es una función de entradas de los instantes de tiempo anteriores, se podría decir que una neurona recurrente tiene en cierta forma memoria [Torres, 2019].

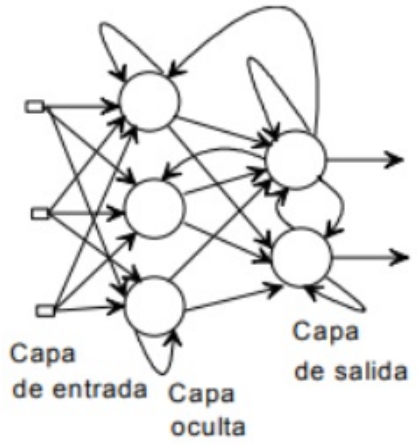


Figura 2.5: Red neuronal recurrente.

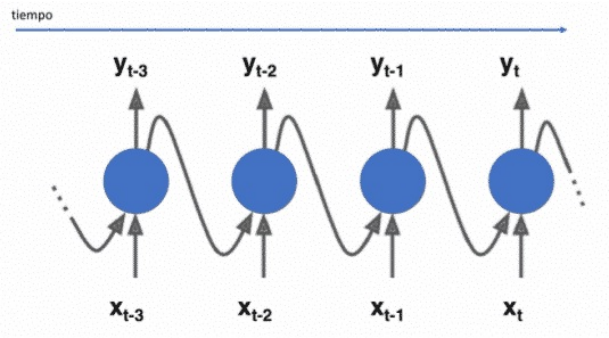


Figura 2.6: Representación temporal de las entradas de la neurona recurrente.

2.5. Aprendizaje supervisado de redes neuronales

Una de las características más importantes de las redes neuronales, como ya se dijo, es su capacidad de aprender automáticamente. Por ello, antes de usarlas en problemas reales, encontramos una fase de aprendizaje supervisado (se parte de un conjunto de datos previamente etiquetados) en la que la red aprende los valores, pesos y sesgo, más adecuados para producir la salida esperada. El proceso de aprendizaje, también llamado entrenamiento, es por tanto un proceso iterativo donde los valores de pesos y sesgo se van refinando paulatinamente con el fin de alcanzar un nivel de operación lo suficientemente bueno.

Hay una serie de hiperparámetros que influyen de forma directa en el progreso del aprendizaje. Estos son los valores que se usarán en las configuraciones utilizadas durante el proceso de entrenamiento, no se obtienen de los datos, por lo que suelen ser indicados. A continuación se enumeran algunos de estos hiperparámetros:

- Función de pérdida o costo. Compara la salida obtenida con la salida deseada, obteniendo el error.

- Optimizador. Algoritmo que controla la actualización de los pesos.

- Tamaño del lote. Número de entradas y salidas que se procesan antes de cada actualiza-

ción de los pesos.

- Número de épocas. Es el número de veces que el sistema procesa el total de datos de entrenamiento.

La correcta selección de los valores que se le fija a los hiperparámetros, anteriormente mencionados, condicionan los resultados finales del proceso de aprendizaje del modelo.

La función de pérdida o costo busca determinar el error entre el valor estimado y el valor real. Se pueden nombrar entre las más usadas a: raíz cuadrada media (RMSE), entropía cruzada categórica (Categorical Cross-Entropy) y la entropía cruzada binaria (Binary Cross-Entropy). La selección adecuada de la función de costo está relacionada directamente con la capacidad que tiene la red para corregir rápidamente los valores de los parámetros (variables) que se estiman durante el proceso de entrenamiento.

El objetivo en el aprendizaje de la red es minimizar la función de costo al encontrar el valor optimizado para los pesos y sesgos (también debemos asegurarnos de que el algoritmo generalice bien, concepto que hace referencia a la capacidad de la red de realizar una predicción correcta de datos que no se vieron antes). El algoritmo que lo permite es el descenso de gradiente, el cual es un algoritmo de optimización iterativo que calcula el gradiente de la función de pérdida con respecto a las variables del modelo. A su vez el método de propagación hacia atrás (Back-propagation), permite entonces que la información del costo fluya hacia atrás a través de las capas de la red para calcular el gradiente [Goodfellow y col., 2016]. Como queremos minimizar el error, modificaremos cada peso en la dirección (negativa) del gradiente como se muestra en la ecuación (2.2).

$$\Theta = \Theta - \eta * \Delta J(\Theta; x, y). \tag{2.2}$$

Θ es el parámetro de peso, η es la tasa de aprendizaje y ΔJ (Θ; x,y) es el gradiente del parámetro de peso Θ

El conjunto de métodos iterativos de reducción de la función de error (búsqueda de un mínimo local), son conocidos cómo los método de optimización basados en el gradiente descendente. Entre los optimizadores más usados podemos citar: **gradiente descendente estocástico** (Stochastic Gradient Descent, SGD) consiste en la introducción de un comportamiento estocástico (aleatorio) limitando el cálculo de la derivada a tan solo una observación por lote, **algoritmo de gradiente adaptativo** (Adaptive Gradient Algorithm, AdaGrad) en él se adapta la tasa de aprendizaje a los parámetros, es decir, en vez de considerar un valor uniforme para todos los pesos se usa una tasa de aprendizaje específica para cada uno realizando actualizaciones más grandes para parámetros poco frecuentes y actualizaciones más pequeñas para parámetros frecuentes, **propagación cuadrática media de la raíz** (Root Mean Square Propagation, RMSprop) también mantiene un factor de entrenamiento diferente para cada dimensión, pero en este caso el escalado del factor de entrenamiento se realiza dividiéndolo por la media del declive exponencial del cuadrado de los gradientes, **estimación adaptativa del momento** (Adaptive moment estimation, Adam) combina las bondades de AdaGrad y RMSProp. Se mantiene un factor de entrenamiento por parámetro y además de calcular RMSProp, cada factor de entrenamiento también se ve afectado por la media del momento del gradiente. El momento es un vector que representa la media en ventana de los anteriores vectores de descenso.

El tamaño del lote influye en el proceso del aprendizaje porque trabajar con un número elevado de muestras puede hacer que el progreso del aprendizaje derive más fácilmente en sobre-entrenamiento (la red aprende de memoria la clasificación de los datos de entrenamiento). Por el contrario, un tamaño reducido del lote ayuda al sistema a generalizar, evitando el sobre-entrenamiento, sin embargo, puede no llegarse a una solución correcta, permaneciendo en un mínimo local de la función de pérdida. lo mismo pudiera ocurrir si el número de épocas es muy alto y los datos no son muy numerosos, puede alcanzarse el sobre-entrenamiento . Sin embargo, con gran cantidad de datos y siendo éstos suficientemente variados y representativos, un número alto de épocas permite, en general, minimizar en mayor medida la función de coste, y por tanto alcanzar un mejor funcionamiento del sistema

En el proceso de aprendizaje al finalizar una época, se procesa un pequeño set de datos no utilizados durante el entrenamiento del modelo, el set de validación, que permite verificar el correcto progreso del aprendizaje y se repite época tras época hasta que los valores se estabilizan (la variación de los parámetros de la red se hace muy pequeña) y el rendimiento de la red converge a un resultado aceptable. Toda vez que el sistema ha sido satisfactoriamente entrenado, los pesos y la estructura de la red quedan fijos, quedando la red neuronal dispuesta para el procesamiento de datos. Una de las principales ventajas de las redes neuronales es que la red en sí aprende la relación existente entre los datos de entrada y salida, adquiriendo la capacidad de generalizar conceptos.

2.6. Redes Neuronales Convolucionales. CNN

La CNN es un tipo de Red Neuronal Artificial que organiza sus capas imitando al corteza visual del cerebro humano para identificar distintas características en las imágenes de entrada. Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía que hacen uso en su mayoría de la operación matemática conocida como convolución, el uso de estas convoluciones en las capas permite al modelo guardar información relacionada con diferentes propiedades presentes en las diferentes partes de las imágenes, tales como contornos o texturas. A su vez, estos elementos extraídos pueden ser también objeto de nuevas convoluciones, de forma que se obtiene información cada vez más abstracta acerca del contenido de las imágenes de estudio para la red neuronal: esto significa que las primeras capas detectan propiedades o formas básicas y se van especializando hasta llegar a capas más profundas capaces de reconocer formas complejas como un rostro o una silueta.

La red neuronal por sí misma ha de entrenarse en una gran cantidad de imágenes para que pueda captar las características únicas de cada objeto y a su vez poder generalizarlo. De esta forma las redes aprenden una representación de los datos, es decir, no solo aprenden a reconocer un objeto sino que también aprenden cuáles son las características importantes que definen de manera única el objeto identificado.

Antes del uso de redes neuronales profundas la exactitud del reconocimiento en imágenes había mejorado a un ritmo lento, pero después de la implementación de redes neuronales profundas la tasa de error en el reconocimiento en imágenes disminuyó del 40 % en el 2010 a aproximadamente 7 % en 2014 y en el 2015 el modelo de red convolucional ResNet152 muestra

sólo un error de 3.57 % que es mejor que la tasa de reconocimiento del humano experto (en ciertos contextos) que es de alrededor del 5 % (Figura.(2.7)) [Alom y col., 2019].

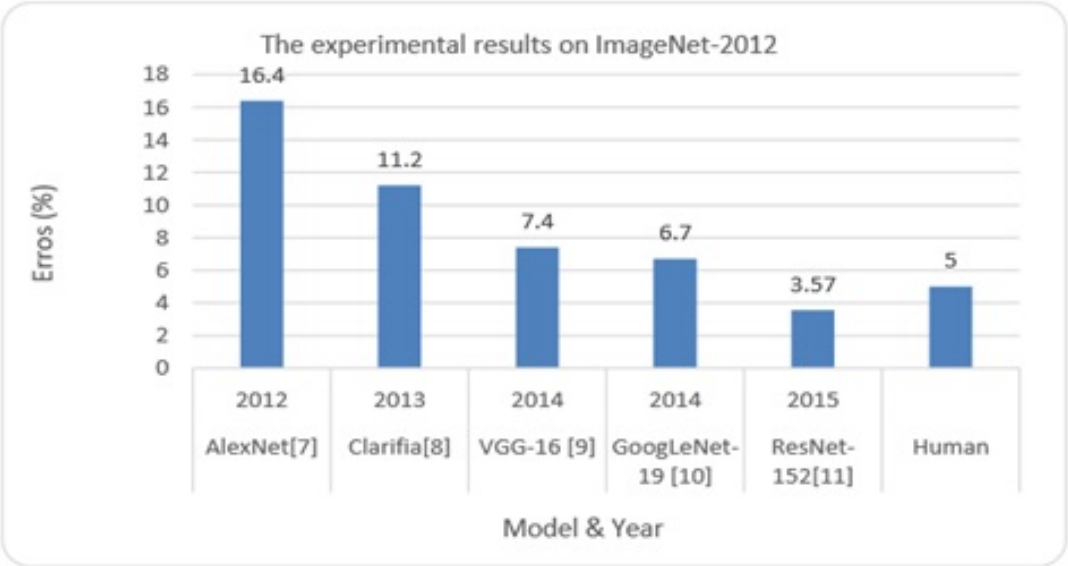


Figura 2.7: Historia de éxito de las técnicas de Deep Learning.

Junto con las capas convolucionales la CNN está formada, generalmente, por una sucesión de capas entre las que podemos nombrar, capas de activación, capas de agrupamiento (Pooling), bien “max pooling” o bien “mean pooling”, capas de normalización por lotes (Batch-Normalization), capas de neuronas clásicas o densas y capas de exclusión (Dropout). Todas en su conjunto dan lugar a una arquitectura que permite, como se dijo anteriormente, extraer las características deseadas de la imagen de entrada y llevar a cabo a partir de ésta, diferentes tareas como clasificación, regresión o segmentación. En función de la tarea final que debe realizar la red, pueden variar algunos de los elementos enumerados e incorporar otros.

2.7. Características de algunas de las capas más usadas en CNN.

2.7.1. Capa Convolucional

La convolución es una forma matemática de combinar dos señales para formar una tercera señal. Es la técnica más importante en el procesamiento de señales digitales [Smith y Ph.D, 2002]. Relaciona la señal de entrada, la señal de salida y la respuesta al impulso. Las matemáticas detrás de la convolución no restringen la duración de estas señales. Sin embargo, especifica la longitud de la señal de salida. que va ha ser igual a la longitud de la señal de entrada, más la longitud de la respuesta al impulso, menos uno [Smith y Ph.D, 2002].

Enmarcándonos en el trabajo con imágenes, la operación de convolución consiste en una vecindad (cada muestra en la salida está influenciada por una región de muestras en la señal

de entrada [Smith y Ph.D, 2002]), un filtro (la respuesta al impulso del sistema, coeficientes de ponderación [Smith y Ph.D, 2002]) y una operación de filtrado definida sobre la vecindad (sumas de entradas ponderadas[Smith y Ph.D, 2002]).

La vecindad de un pixel se define como la relación que tiene un pixel de manera posicional con los pixeles mas cercanos a él, dentro del procesamiento de imagenes las vecindades 4-vecinos y 8-vecinos son las más comunes, con nombre Von Neuman y Moore respectivamente [Rojas, 2018]. La operación crea un pixel con las coordenadas del centro de la vecindad y cuyo valor depende de la operación de filtrado, de este modo la imagen filtrada es generada con el centro del filtro visitando cada pixel de la imagen de entrada [Gonzalez y Woods, 2008].

Generalmente en las capas convolucionales el núcleo previamente nombrado es una matriz cuadrada de tamaño impar y menor a la imagen de entrada e influye directamente en el resultado del filtrado, cómo se dijo anteriormente, en base a dos aspectos, sus dimensiones (n × n) y los valores que contiene. El primero, las dimensiones, lo define el diseñador cuando configura la arquitectura de la red, mientras que los valores que contiene, que se denominan pesos, son ajustados automáticamente durante el proceso de entrenamiento.

En la convolución se realizan los siguientes pasos, la operación se realiza sobre un fragmento de la imagen original del mismo tamaño al del núcleo: [Martín López, 2019]

1. Se voltea el núcleo tanto horizontal como verticalmente. Si el núcleo es simétrico, este paso no tiene ninguna consecuencia.

2. Se multiplica cada elemento de la matriz núcleo por su similar local de la imagen, estamos refiriéndonos a los píxeles que conforman la vecindad del píxel central y que están determinados por el tamaño del filtro.

3. Se suman todos los resultados de las multiplicaciones del paso anterior.

4. El elemento central del fragmento tomado de la imagen obtendrá en el valor de salida el resultado que se obtuvo en el paso 3.

5. Este proceso se va repitiendo, colocando el centro del núcleo sobre cada elemento de la imagen.

Como se aprecia, de todo lo anterior, la operación de convolución se trata de realizar una combinación lineal sobre cada ventana de n×n píxeles de la entrada. (Figura. (2.8))

Las ecuaciones que establecen la forma de calcular el valor de cada píxel de salida serían entonces: (2.9)

Algo que nos puede interesar al convolucionar es mover el filtro con un paso (stride) mayor a uno, con lo cual logramos una salida más pequeña, en lo que se conoce como downsampling. Otra característica esencial de cualquier implementación de red convolucional es la condición de llenar con ceros los píxeles que completan la vecindad de los píxeles del borde de la imagen, Esto se denomina **padding**. Hay ,principalmente, dos casos de padding que se suelen trabajar: el primero es no utilizar padding, permitiendo que el filtro visite solamente aquellas posiciones para las cuales está contenido completamente el filtro (suele denominarse convolución válida y el resultado es una matriz de menor tamaño), un segundo caso es aquel en el que se mantiene el tamaño de la salida igual al de la entrada (convolución “same”).

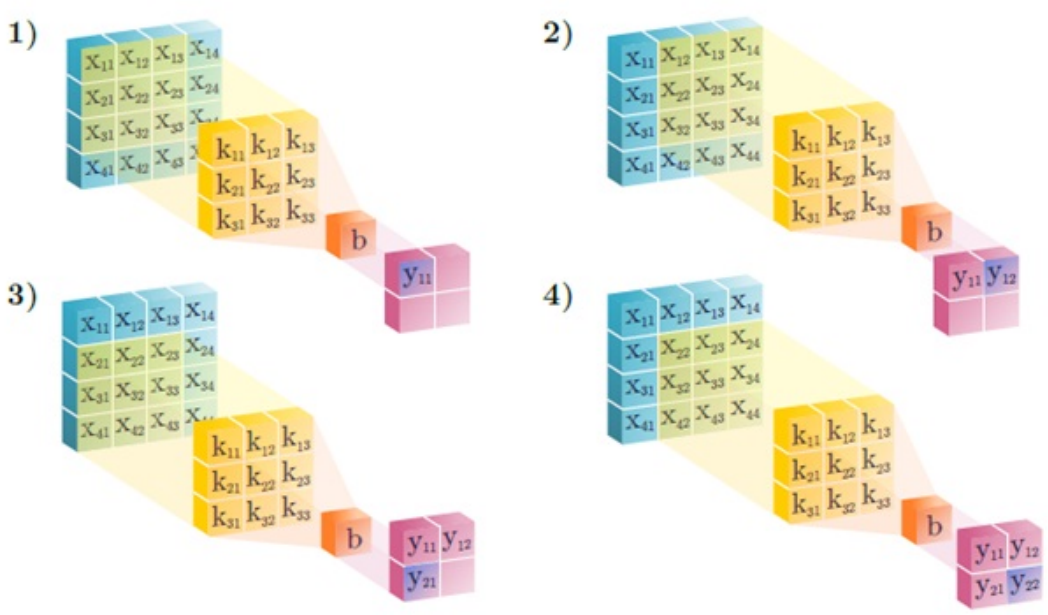


Figura 2.8: Esquema de la operación convolución sobre una imagen (x) de dimensiones 4 × 4, empleando un kernel (k) de dimensiones 3 × 3.

$$y_{11} = \sum_{j=1}^{3}\sum_{i=1}^{3} x_{i,\,j}\; k_{i,\,j} + b \qquad y_{12} = \sum_{j=1}^{3}\sum_{i=1}^{3} x_{i,\,j+1}\; k_{i,\,j} + b$$

$$y_{21} = \sum_{j=1}^{3}\sum_{i=1}^{3} x_{i+1,\,j}\; k_{i,\,j} + b \qquad y_{22} = \sum_{j=1}^{3}\sum_{i=1}^{3} x_{i+1,\,j+1}\; k_{i,\,j} + b$$

Figura 2.9: Ecuaciones para calcular el valor de cada píxel de salida en la operación de convolución sobre una imagen (x) de dimensiones 4 × 4, empleando un kernel (k) de dimensiones 3 × 3.

El parámetro sesgo, del que se habló con anterioridad, también es entrenable y constituye un valor adicional (offset) que se aplica a todos los píxeles de salida de la convolución.

Al resultado producido por un filtro de convolución se le llama imagen de características, dado que ofrece información sobre la presencia o ausencia de la característica buscada por el filtro en cuestión.

En el contexto de redes neuronales, cuando nos referimos a convolución se está pensando en la aplicación de varias convoluciones en paralelo. Ya vimos antes que cada filtro es capaz de extraer solo una característica, de modo que lo que nos interesa es combinar varios de estos filtros. Es por esto que es conveniente pensar al filtro como una sola entidad de p × q × r, donde p y q representan las filas y columnas del filtro y r el número de filtros que estamos utilizando. Lo que tenemos entonces es que nuestro filtro es un tensor de dimensión tres. Esto corresponde al caso de que la entrada sea 2D, un solo canal (por ejemplo una imagen sólo en tonos de gris); pero nuestra entrada suele contar con varios canales (caso de una imagen a color, RGB). Entonces no tenemos solo una matriz con las intensidades presentes en cada

píxel, sino que tenemos tres de estas imágenes, una por cada canal de color (rojo, verde, azul). De esta manera, se puede pensar a la imagen como un tensor 3D, y en consecuencia nuestro filtro pasa a ser 4D, ya que se necesita un filtro distinto por cada canal de entrada.

Dado que se cuenta con N filtros de convolución, se generarán N imágenes de características por cada capa convolucional de la red.

El usar muchas capas en la CNN le aporta la propiedad de aprender jerarquías espaciales de patrones porque el mundo visual es fundamentalmente jerárquico espacialmente [Goodfellow y col., 2016]. De manera general se espera que la primera capa de convolución aprenda pequeños patrones locales como son los bordes de un objeto, la segunda capa a encontrar patrones más elaborados a partir de los bordes, la siguiente capa a encontrar partes de objetos, la siguiente objetos y así sucesivamente, lo que permite que las redes neuronales convolucionales aprendan conceptos visuales complejos.

2.7.2. Capa ReLU

Las capas de convolución típicamente van seguidas de una capa ReLU, nombrada así por el uso de la función de activación de igual nombre que vimos con anterioridad.

La salida generada por una capa de convolución, al igual que la de la neurona de un perceptrón, es una salida lineal, dado que todas las operaciones internas son operaciones lineales. La capa ReLU realiza una operación de umbral para cada elemento, de forma que cualquier valor de entrada menor que cero se iguala a cero. Este tipo de capa no modifica el tamaño de su entrada y adiciona la no-linealidad en la red.

Si no se aplicasen no linealidades el hecho de añadir nuevas capas ocultas al perceptrón no tendría ninguna utilidad, dado que la combinación lineal de N funciones lineales sigue devolviendo una función lineal. Es decir, seguiría siendo imposible generar fronteras de decisión no lineales que clasificasen correctamente conjuntos de datos no linealmente separables.

Existe la posibilidad de utilizar esta función con algunas variantes como, por ejemplo, filtrando las entradas negativas a la salida y multiplicando los valores de entrada menores de 0 por un escalar fijo, capa ReLU con fugas [He y col., 2015b], o estableciendo para cualquier valor negativo de x, un valor realmente pequeño de x, denominada Leaky ReLU [Maas y col., 2013].

2.7.3. Capa Pooling

Las capas de convolución representan la mayor parte del coste computacional empleado por la red debido al número tan elevado de operaciones matemáticas que en ellas se realizan y la memoria que consumen. A fin de reducir este coste se introduce la capa 'pooling' o reducción. Esta reducción de tamaño significará una pérdida de información de la imagen. No obstante, dado que es cierto que la localización exacta de una característica es menos importante que su localización aproximada con respecto a otras características, la capa de reducción de muestreo permite [Pastor Martín, 2018]:

- reducir paulatinamente el consumo espacial de la representación de la imagen,

- reducir el número de neuronas (los mapas de características) y el costo computacional de la red,

- reducir el sobreajuste y aumentar la invariancia frente a la traslación.

Los métodos de agrupación más utilizados son agrupación máxima o 'max pooling', agrupación promedio o 'average pooling' y agrupación L2 o 'L2-norm pooling', destacando por ser la más utilizada la técnica de agrupación máxima (Figura. (2.10)).

Al usar 'max pooling' dividimos la imagen de entrada en un conjunto de rectángulos y respecto a cada subregión tomamos el valor máximo de activación de la región seleccionada, ésta operación de reducción de muestreo actúa de forma independiente en cada capa de la entrada.

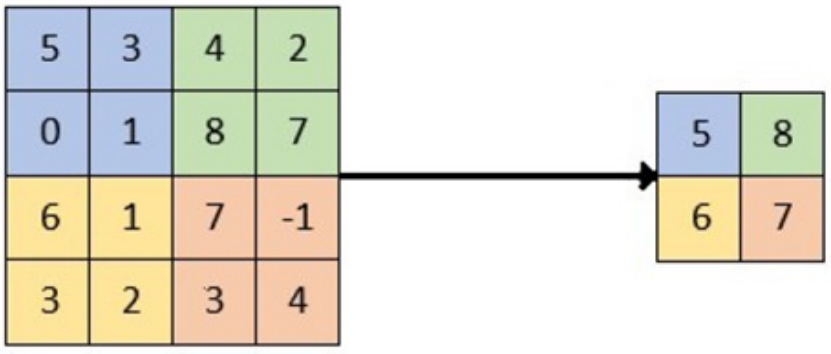


Figura 2.10: Esquema de la operación de 'max pooling' con filtros 2×2 y con salto o stride de 2 elementos.
Se reduce el muestreo en un 75 % de las activaciones al realizar la operación sobre 4 elementos.

2.7.4. Capa Dropout o de Exclusión

La capa de exclusión o dropout tiene la función de intentar evitar el problema de sobreajuste en la red, de forma que los pesos y sesgos de las capas convolucionales y totalmente conectadas no se ajusten en exceso a las imágenes de entrenamiento provocando un mal funcionamiento de la red al evaluar en ella las imágenes de prueba [Srivastava y col., 2014].

El proceso seguido en este tipo de capa es eliminar un conjunto aleatorio de activaciones con una probabilidad dada estableciendo sus valores a cero. El abandono de las entradas de una capa pueden ser variables de entrada en la muestra de datos o activaciones de una capa anterior. Tiene el efecto de simular una gran cantidad de redes con estructuras de red muy diferentes y, a su vez, hace que los nodos de la red sean generalmente más robustos a las entradas [Srivastava y col., 2014].

La capa de exclusión se puede utilizar después de capas convolucionales y después de capas de muestreo o agrupamiento aunque a menudo sólo se usa después de las totalmente conectadas.

2.7.5. Capa Batch-Normalization

Una técnica que se emplea mucho en Deep Learning para agilizar el proceso de aprendizaje es la de Batch Normalization, consiste en estandarizar/normalizar las entradas a cada capa de la red por cada lote [Haykin, 2009]. Cuando normalizamos los datos solo la capa de entrada se beneficia de esto, conforme los datos pasan por otras capas ocultas esta normalización se va perdiendo. El método de batch normalization normaliza los datos antes de que pasen por la función de activación en cada capa de la red neuronal.

Los datos de entrada de una capa de la red se pueden expresar como una matriz de cuatro dimensiones (m, f, p, q), donde m es el tamaño del lote, f es el número de mapas de características, y p y q son el ancho y alto de estos mapas respectivamente. En la normalización por lotes se trata cada mapa de características como un proceso de características (una neurona). Atendiendo a esto la normalización por lote se divide principalmente en 4 pasos:

-Calcular la media de cada lote de datos de entrenamiento. Ecuación (2.3).

-Calcular la varianza de cada lote de datos de entrenamiento. Ecuación (2.4).

-Usar la media y la varianza obtenidas para normalizar el lote de datos de entrenamiento para obtener una distribución media igual 0 y una varianza igual 1. Ecuación (2.4).

-Transformación de escala y desplazamiento. Se introducen dos parámetros γ y β, γ para ajustar el valor, más β para aumentar el desplazamiento estos dos parámetros son aprendidos por la red durante el entrenamiento. Ecuación (2.6).

$$\mu_\beta = 1/m \sum_i^m x_i. \tag{2.3}$$

$$\sigma_\beta^2 = 1/m \sum_i^m .(x_i - \mu_\beta)^2. \tag{2.4}$$

$$x_i^{norm} = \frac{X_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}. \tag{2.5}$$

$$y_i = \gamma x_i^{norm} + \beta = BN_{\gamma,\beta}(x_i). \tag{2.6}$$

Entrenar redes neuronales profundas, por ejemplo, redes con decenas de capas ocultas, es un desafío. Un aspecto de este desafío es que el modelo se actualiza capa por capa hacia atrás desde la salida hasta la entrada usando una estimación de error que asume que los pesos en las capas antes de la capa actual son fijos. Debido a que todas las capas se cambian durante una actualización, el procedimiento de actualización siempre persigue un objetivo en movimiento [Goodfellow y col., 2016].

Este continuo cambio en la distribución de las activaciones ralentiza considerablemente el entrenamiento ya que cada capa debe aprender a adaptarse a nuevas distribuciones de datos cada vez. La solución de realizar Batch Normalization tiene como efecto estabilizar el proceso

de aprendizaje y reducir dramáticamente el número de épocas requeridas para entrenar una red profunda.

Como efecto beneficioso añadido, la capa 'batch normalization' también mejora el flujo del gradiente a través de la red, al hacerlo menos dependiente de la escala de los parámetros. Si se combina con funciones de activación como la función ReLU, esta capa evita que el flujo de información quede interrumpido cuando los valores producidos son negativos, dado que desplazará de nuevo estos valores, fijados a cero por la función de activación [Rodriguez, 2020].

2.7.6. Capa Fully Connected o Totalmente Conectada

Las capas fully connected, totalmente conectadas o capa densa, como su nombre indica, conectan todas las neuronas de la capa anterior con las entradas de la siguiente. Normalmente son utilizadas al final de las capas convolucionales.

Las salidas obtenidas de las capas convolucionales representarán características de alto nivel de la imagen de entrada. El objetivo de esta capa es utilizar estas características combinándolas para clasificar la imagen de entrada en distintas clases en función de los datos obtenidos. Es usual usar en los problemas de clasificación multiclase una capa densa de salida con la función de activación 'softmax'. La función softmax es una generalización de la regresión logística que puede ser aplicada a datos continuos. Una de sus principales características es que está normalizada, es decir, la suma de las salidas de la función softmax para cada uno de los componentes del vector da como resultado la unidad. En términos generales, esta función calculará las probabilidades de cada clase objetivo sobre todas las clases objetivo posibles. La ecuación (2.7) muestra su expresión:

$$\mathbf{f}(\mathbf{X_i}) = \frac{e^{X_i}}{\sum_{k=1}^{N} e^{X_k}}. \tag{2.7}$$

La fórmula calcula la exponencial del valor de entrada dado y la suma de los valores exponenciales de todos los valores en las entradas. Luego, la relación de la exponencial del valor de entrada y la suma de los valores exponenciales es la salida de la función softmax.

En la gráfica (Figura. (2.11)) puede observarse cómo para un vector de ejemplo formado por 6 elementos, la suma de las salidas de la función softmax es igual a 1.

2.8. Filtros digitales

Los filtros digitales constituyen uno de los principales modos de operar en el procesamiento de imágenes digitales. Se trata de métodos para resaltar o suprimir, de forma selectiva, información contenida en una imagen a diferentes escalas espaciales, para destacar algunos elementos de la imagen, o también para ocultar valores anómalos.

Los principales objetivos que se persiguen con la aplicación de filtros son [EcuRed, 2009]:

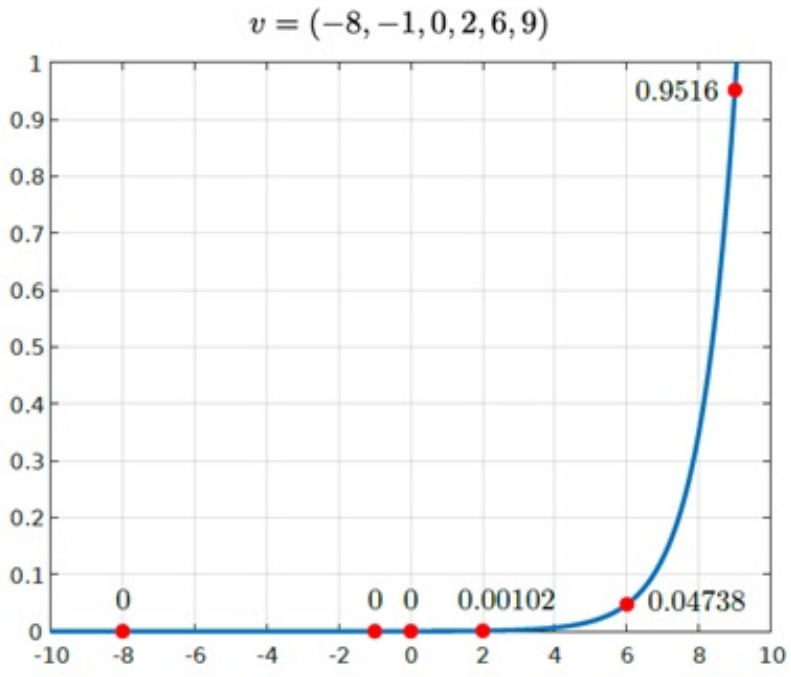


Figura 2.11: Representación gráfica de la función Softmax.

$$\sum_i f(v) = 0 + 0 + 0 + 0.00102 + 0.04738 + 0.9516 = 1$$

• Suavizar la imagen: reducir la cantidad de variaciones de intensidad entre píxeles vecinos.

• Eliminar ruido: eliminar aquellos píxeles cuyo nivel de intensidad es muy diferente al de sus vecinos y cuyo origen puede estar tanto en el proceso de adquisición de la imagen como en el de transmisión.

• Realzar bordes: destacar los bordes que se localizan en una imagen.

• Detectar bordes: detectar los píxeles donde se produce un cambio brusco en la función intensidad.

Por tanto, se consideran los filtros como operaciones que se aplican a los píxeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella. El proceso de filtrado puede llevarse a cabo sobre los dominios de frecuencia y/o espacio.

Los filtros de frecuencia procesan una imagen trabajando sobre el dominio de la frecuencia en la Transformada de Fourier de la imagen. Para ello, ésta se modifica siguiendo el Teorema de la Convolución correspondiente:

1. Se aplica la Transformada de Fourier.

2. Se multiplica posteriormente por la función del filtro que ha sido escogido.

3. Para concluir re-transformándola al dominio espacial empleando la Transformada Inversa de Fourier.

Como la multiplicación en el espacio de Fourier es idéntica a la convolución en el dominio espacial, todos los filtros podrían, en teoría, ser implementados como un filtro espacial.

Existen básicamente tres tipos distintos de filtros que pueden aplicarse [EcuRed, 2009]:

• Filtro paso bajo: atenúa las frecuencias altas y mantiene sin variaciones las bajas. El resultado en el dominio espacial es equivalente al de un filtro de suavizado, donde las altas frecuencias que son filtradas se corresponden con los cambios fuertes de intensidad. Consigue reducir el ruido suavizando las transiciones existentes.

• Filtro paso alto: atenúa las frecuencias bajas manteniendo invariables las frecuencias altas. Puesto que las altas frecuencias corresponden en las imágenes a cambios bruscos de densidad este tipo de filtros, ofrece mejoras en la detección de bordes en el dominio espacial, ya que estos contienen gran cantidad de dichas frecuencias, es decir, refuerzan los contrastes que se encuentran en la imagen.

• Filtro paso banda: atenúa frecuencias muy altas o muy bajas manteniendo una banda de rango medio.

La investigación que aborda este trabajo hace uso de las características que fijan sobre la imagen los filtros paso bajo, de los cuales existen varias posibilidades [“Filtros digitales”, 2003]:

• Filtro de la media: asigna al pixel central la media de todos los pixeles incluidos en la ventana. La matriz de filtrado estaría compuesta por unos divididos entre el número total de elementos en la matriz.

• Filtro de media ponderada: los elementos de la matriz de filtrado no son todos 1 sino que se da más peso a uno de ellos (generalmente el central) para obtener un resultado más parecido a la imagen original y evitar que aparezca borrosa.

• Filtro de la mediana: tiene la ventaja de que el valor final del pixel es un valor real presente en la imagen y no un promedio, de este modo se reduce el efecto borroso que tienen las imágenes que han sufrido un filtro de media. Además el filtro de la mediana es menos sensible a valores extremos. El inconveniente es que resulta más complejo de calcular ya que hay que ordenar los diferentes valores que aparecen en los pixeles incluidos en la ventana y determinar cuál es el valor central (Figura (2.12)).

• Filtros adaptativos: Son considerablemente más complejos ya que los coeficientes de ponderación se recalculan para cada uno de los pixeles en función del histograma.

• Filtros gaussianos: Simulan una distribución gaussiana bivariante. Similar al filtro de la media provoca: disminución de la nitidez, aumento de borrosidad, pérdida de detalles.

En esta investigación, específicamente, usamos el filtro gaussiano en el entrenamiento de los modelos de redes neuronales con los que trabajamos.

2.9. Filtro Gaussiano

Descrito por la función gaussiana. Produce un suavizado más uniforme que el de la media El valor máximo aparece en el pixel central y disminuye hacia los extremos tanto más rápido

cuanto menor sea el parámetro de desviación estándar sigma (σ). El resultado será un conjunto de valores entre 0 y 1. La ecuación para calcularlo es: (Ecuación (2.8))

$$\mathbf{G(x,y)} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.8}$$

El efecto del suavizado gaussiano es desdibujar una imagen. El grado de suavizado se determina a través del valor de la desviación estándar.

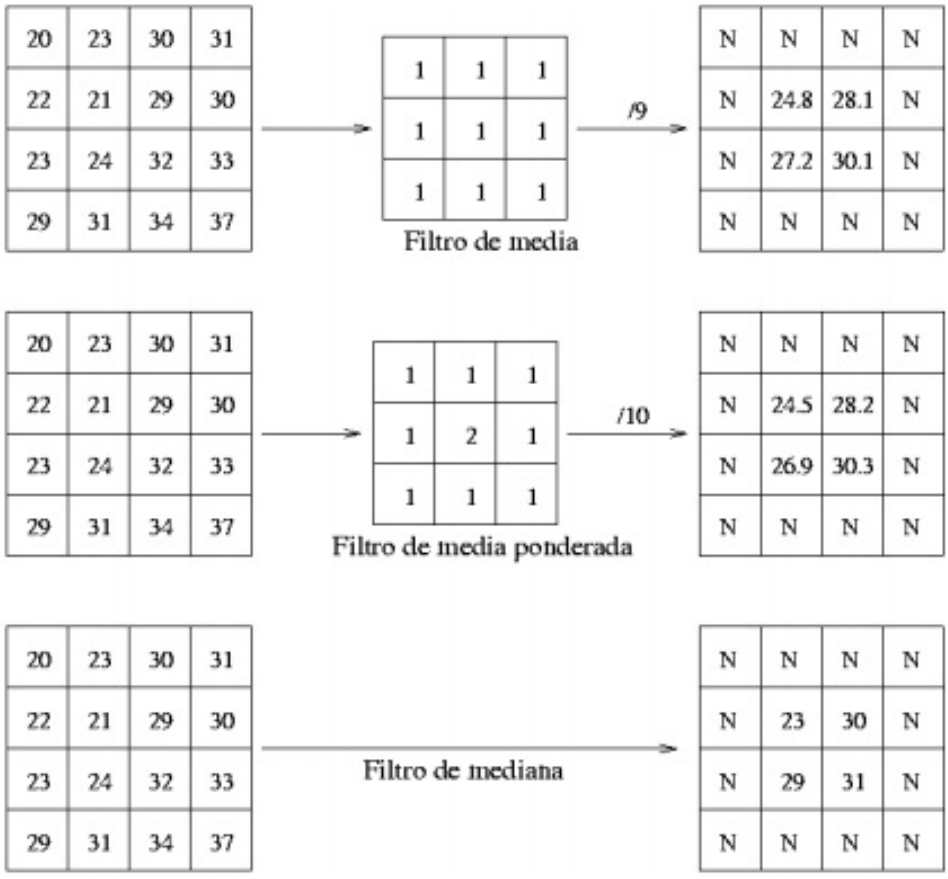


Figura 2.12: Tipos de filtro paso bajo y mecanismos de aplicación.

La salida del filtro gaussiano es un promedio pesado de cada píxel y su vecindad, siendo el de mayor peso el píxel central. Esto lo diferencia del filtro de media donde todos los pesos son iguales, se puede ver un ejemplo de lo anterior en la Tabla 2.1.

Como se dijo anteriormente, el parámetro sigma que figura en la función gaussiana es la desviación estándar y permite modular la intensidad del suavizado gaussiano. Cuanto mayor es σ más aplastada será la campana y, por tanto, mayor será su efecto de suavizado [Barrios, 2004] (Fig. (2.14)(2.15)).

0.00296902	0.01330621	0.02193823	0.01330621	0.00296902
0.01330621	0.0596343	0.09832033	0.0596343	0.01330621
0.02193823	0.09832033	0.16210282	0.09832033	0.02193823
0.01330621	0.0596343	0.09832033	0.0596343	0.01330621
0.00296902	0.01330621	0.02193823	0.01330621	0.00296902

Tabla 2.1: Valores de una matriz de peso de un filtro gaussiano de dimensión 5×5 y σ=1.

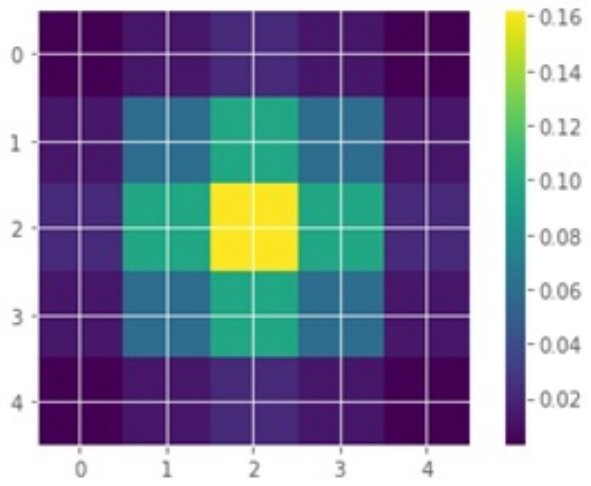


Figura 2.13: Representación de la matriz de peso, el valor máximo en el pixel central y disminuye hacia los extremos.

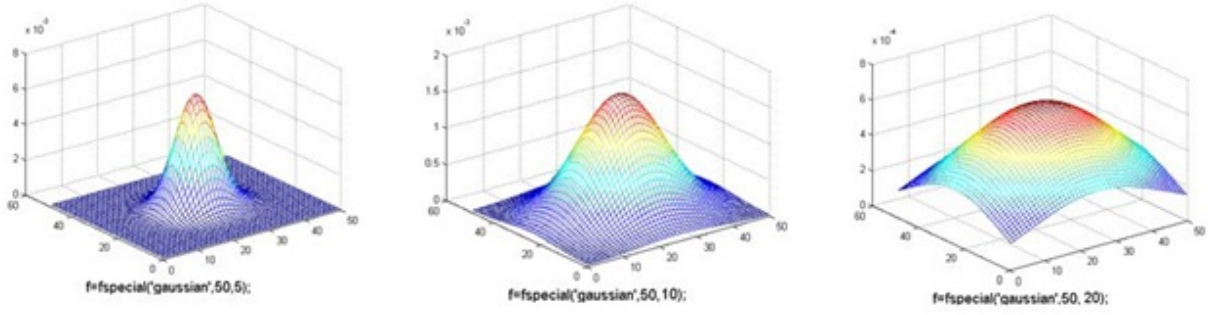


Figura 2.14: Representación de filtros gaussianos con sigmas 5, 10 y 20.

2.10. Convolución 2D y Convolución Espacial Separable

Uno de los principales problemas que presentan las capas convolucionales es su alta carga computacional, como se menciona anteriormente, ya que deben realizar numerosas multiplicaciones. En una capa convolucional clásica en la que la entrada tiene M canales, la salida tiene dimensiones (Y×Y), y se aplican N kernels de dimensiones (K×K), el número de multiplicaciones que se llevan a cabo en dicha capa convolucional se calcula como indica la

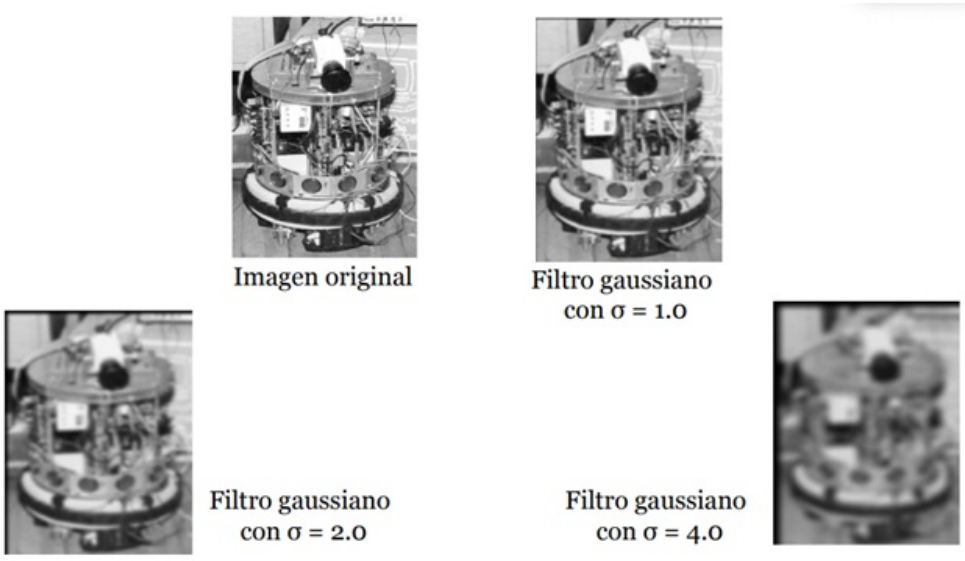


Figura 2.15: Imágenes que resultan de aplicar filtro gaussiano con diferentes valores de σ.

ecuación (2.9), entendiendo un valor de salto o stride igual a 1.

$$\mathbf{N_{0,(multiplicaciones)}} = K^2 * Y^2 * MN. \tag{2.9}$$

Hagamos una convolución de (5×5) en una imagen de tres canales (RGB) de tamaño 12x12 sin relleno y un stride de 1 para ilustrar el resultado de esta ecuación [Wang, 2018].

- el kernel de 5×5 se somete a una multiplicación escalar con cada 25 píxeles, dando 1 número cada vez. Terminamos con una imagen de 8×8 píxeles, ya que no hay relleno (12–5+1=8) (dimensión de la entrada menos la de kernel más el stride); pero debido a que la imagen tiene 3 canales, nuestro núcleo convolucional también debe tener 3 canales y esto significa que, en lugar de hacer 5×5 = 25 multiplicaciones, en realidad hacemos 5×5×3 = 75 multiplicaciones cada vez que se mueve el núcleo.

Después de pasar por un kernel de 5×5×3, la imagen de 12×12×3 se convertirá en una imagen de 8×8×1. (Figura (2.16))

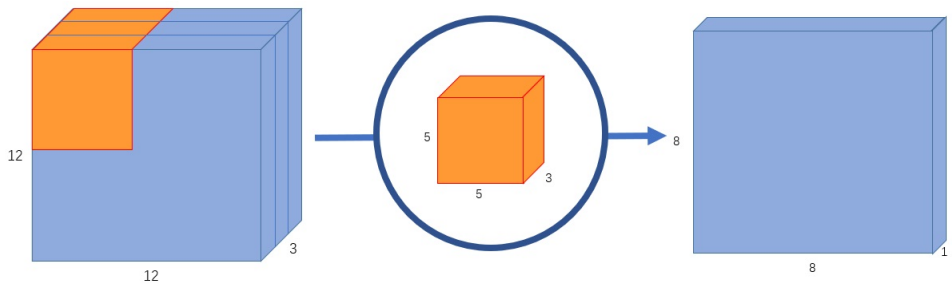


Figura 2.16: Convolución con salida de 8×8×1.

Si queremos aumentar el número de canales de salida de nuestra imagen, para lograr la separación de características, como sucede en una capa convolucional, debemos crear tantos

núcleos como canales de salida queremos en nuestra imagen. De esta forma para crear una imagen de salida de 8×8×256 tendremos 256 núcleos donde cada uno crea una imagen de 8×8×1, luego al apilarlos juntos tendremos una señal de salida de 8×8×256 (Figura (2.17)).

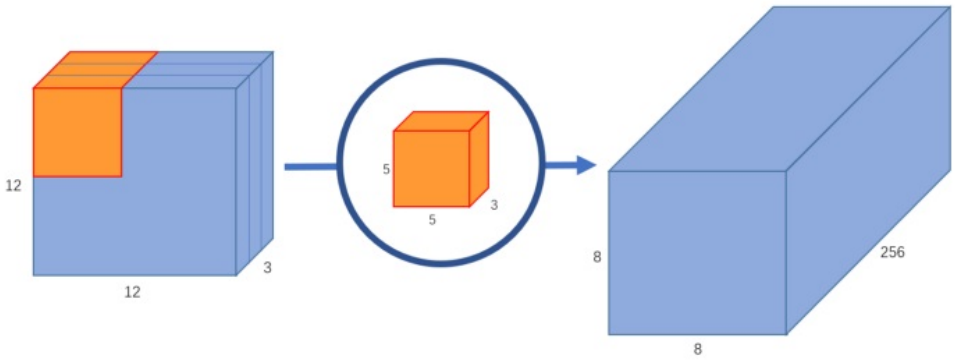


Figura 2.17: Convolución con salida de 8x8x256.

Calculando el número de operaciones en la convolución tenemos, según la ecuación (2.9):

Número de multiplicaciones = (5×5) × (8×8) × 3 × 256 = 1,228,800

Con el objetivo de reducir el número de multiplicaciones y por tanto reducir la carga computacional, aumentando la velocidad y mejorando el rendimiento, se desarrolló una variante de la convolución clásica, denominada Convolución Separable, en inglés Depthwise Separable Convolution [Chollet, 2017].

Su estructura interna se divide en dos etapas claramente diferenciables:

- Etapa 1: denominada en inglés Depthwise Convolution, convolución en profundidad, realiza una convolución individual de cada canal de la entrada con un kernel diferente, de dimensiones según diseño, pero siendo siempre las dimensiones de estos kernels iguales. Si la entrada tiene M canales, hay M kernels y cada uno se convoluciona de forma individual con cada canal de la entrada, de forma que la salida tiene M canales también.(Figura (2.18))

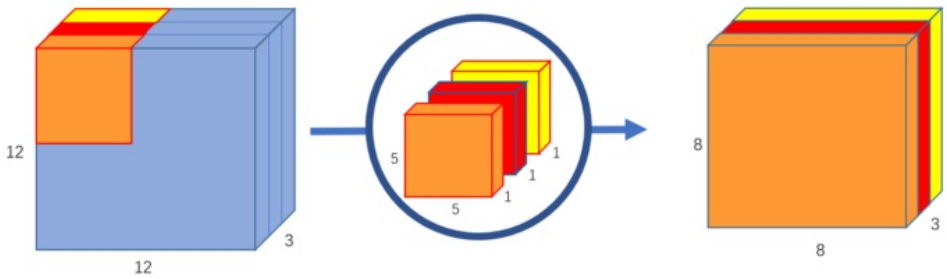


Figura 2.18: Convolución en profundidad.
En el ejemplo se utilizan 3 núcleos para transformar una imagen de 12×12×3 en una imagen de 8×8×3.

- Etapa 2: denominada en inglés Pointwise Convolution o convolución puntual, puede entenderse como una capa convolucional clásica en la que la entrada es la salida de la etapa

anterior, con la particularidad de que los kernels (cada uno con el número de canales que tenga la salida de la Etapa 1) tienen dimensiones 1×1, siendo el número de kernels el especificado por el diseñador (Figura (2.19)).

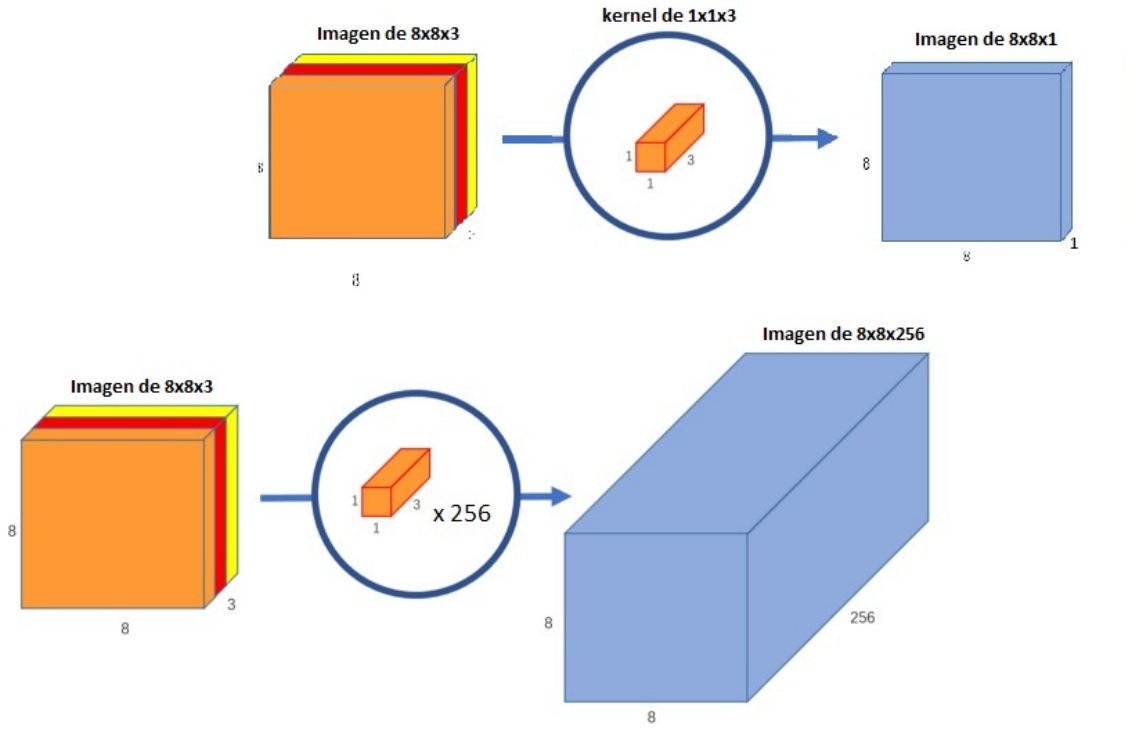


Figura 2.19: Convolución puntual.
Convolución con 256 núcleos, generando una imagen de salida con 256 canales.

- Calculando el número de operaciones, en la convolución en profundidad, tenemos 3 kernels de 5×5×1 que se mueven 8×8 veces. Eso es 3×5 ×5×8×8 = 4800 multiplicaciones.

- Calculando el número de operaciones, en la convolución puntual, tenemos 256 núcleos de 1 ×1×3 que se mueven 8×8 veces. Eso es 256×1×1×3 ×8×8 = 49,152 multiplicaciones.

Si sumamos ambas convoluciones:

Número de multiplicaciones = 53 952 mucho menos que 1.228.800.

La principal diferencia es esta: en la convolución normal, estamos transformando la imagen 256 veces. Y cada transformación usa hasta 5×5×3×8×8 = 4800 multiplicaciones. En la convolución separable, solo se transforma la imagen una vez - en la convolución en profundidad. Luego, la imagen transformada es representada en 256 canales. Sin tener que transformar la imagen una y otra vez.

Vista desde fuera, una Convolución Separable se utiliza del mismo modo que una Capa Convolucional convencional. Se especifica el número de kernels a aplicar y las dimensiones de éstos, así como el valor del stride. Las dimensiones de la salida coinciden con las que tendría en el caso de utilizar una Capa Convolucional convencional.

Si se compara el resultado de las operaciones para los dos tipos de convolución se obtiene la relación de reducción del número de multiplicaciones, que puede aproximarse a la relación de mejora en carga computacional. Por otro lado, existe también una mejora en el número de pesos que contiene una capa convolucional separable. La expresión de dicha relación coincide con la de mejora del rendimiento.

En el marco de este trabajo la posibilidad que brinda el uso de la capa de convolución separable en su primera etapa, depthwise convolution, toma particular relevancia porque es el funcionamiento de este método el que permite operar con los filtros pasabajo a la salida de cada canal de una capa convolucional clásica y de esta forma establecer una de las condiciones que exige un entrenamiento curricular por suavizado.

En esta investigación buscaremos aprovechar las ventajas que deja en nuestra tarea de clasificación usar redes previamente preentrenadas con bases de datos grandes. Este mecanismo conocido cómo Transferencia de Aprendizaje se explicará con detalles en próximas secciones, por ahora valga solo introducir que para este propósito es necesario que la red preentrenada haya sido entrenada para resolver un problema de carácter mas general, del que nuestro problema se pueda considerar un caso particular. Es por ello que en nuestro trabajo escogeremos dos modelos de redes neuronales convolucionales, del estado del arte, que evidenciaron resultados muy buenos en la tarea de clasificar la base de datos ImageNet , una base de datos de imágenes con 1000 categorías etiquetadas manualmente (para investigación de visión artificial). El conjunto de entrenamiento , en esta base de datos, es de 1,2 millones, el conjunto de verificación es 50.000 y el conjunto de prueba es 100.000 [Russakovsky y col., 2015]. En términos de clasificación de imágenes, la precisión de la competencia ImageNet se ha utilizado como punto de referencia para los algoritmos de clasificación de visión por computadora. Otro aspecto importante es que la red (modelo y pesos entrenados) esté diponible en Kera, librería lenguaje informático Python, que usaremos para la implementación del método de entrenamiento en los redes usadas.

2.11. Modelo de red neuronal convolucional VGG16

El modelo de red neuronal convolucional VGG16 fue propuesto por K. Simonyan y A. Zisserman de la Universidad de Oxford en el artículo, Very Deep Convolutional Networks for Large-Scale Image Recognition (Redes convolucionales muy profundas para el reconocimiento de imágenes a gran escala) [Simonyan y Zisserman, 2014]. Fue uno de los modelos ganadores presentados al Desafío de reconocimiento visual a gran escala 2014 [Russakovsky y col., 2015]. El mismo realiza una mejora sobre un modelo de red anterior llamado AlexNet [Krizhevsky y col., 2012] y propuso reemplazar los filtros de gran tamaño de kernel (11×11 y 5×5 en la primera y segunda capa convolucional de ése modelo) con múltiples filtros de tamaño de kernel de 3×3 uno tras otro.

La entrada a la primera capa convolucional del modelo VGG16 es una imagen RGB de tamaño fijo 224×224. Los filtros usados, debido a su dimensión, la menor posible para capturar la noción de izquierda / derecha, arriba / abajo, centro; poseen un campo receptivo muy pequeño. El paso de convolución, el stride, se fija en 1 píxel; el padding en la entrada de

la capa es tal que la resolución espacial se conserva después de la convolución, es decir, el relleno es de 1 píxel. La agrupación espacial, el pooling, se lleva a cabo mediante cinco capas de agrupación máxima, que siguen algunas de las capas convolucionales, no todas van seguidas de la agrupación máxima y esta se realiza en una ventana de 2×2 píxeles. Tres capas totalmente conectadas constituyen la cabeza de la red, las dos primeras tienen 4096 perceptrones cada una, la tercera realiza una clasificación de 1000 clases y, por lo tanto, contiene 1000 perceptrones (uno para cada clase). La última usa la función de activación softmax. Todas las capas ocultas hacen uso de la función de activación ReLU [Muneeb ul, 2018].

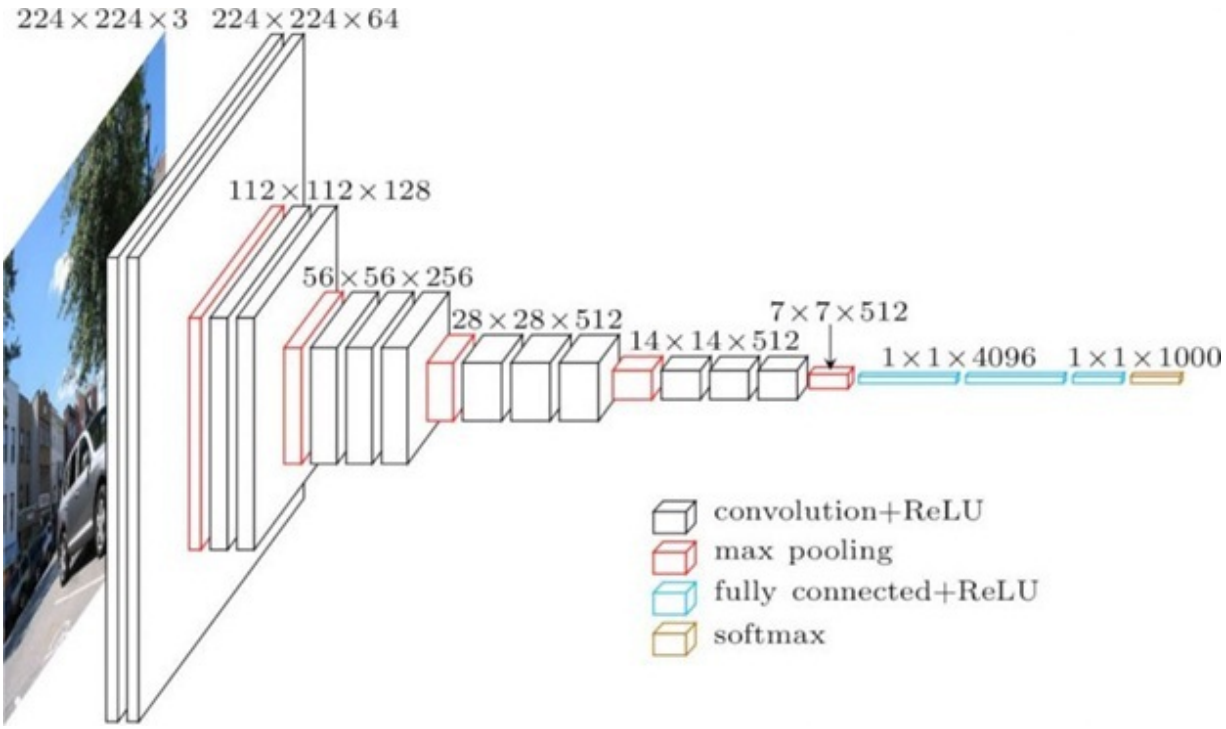


Figura 2.20: Arquitectura del modelo de red VGG16.

El modelo de red VGG16 supera significativamente a la generación anterior de modelos en las competencias ILSVRC-2012 e ILSVRC-2013 [*ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. 2014]. El resultado de VGG16 también compite por el ganador de la tarea de clasificación GoogLeNet con un 6,7 % de error [*ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. 2014] y supera sustancialmente a la presentación ganadora del ILSVRC-2013 [*ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. 2014], Clarifai, que logró un 11,2 % con datos de entrenamiento externos y un 11,7 % sin ellos.

Con respecto al rendimiento de red única, la arquitectura VGG16 logra el mejor resultado (error de prueba del 7,0 %), superando a GoogLeNet en un 0,9 % [*ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. 2014]. Demostrando que la profundidad del modelo es beneficiosa para la precisión de la clasificación y que se puede lograr un rendimiento de vanguardia en el conjunto de datos de desafío de ImageNet utilizando una arquitectura ConvNet convencional con una profundidad sustancialmente mayor.

2.12. Modelo de red neuronal basado en bloques residuales, ResNet

La redes residuales se inspiran en el hecho biológico de que algunas neuronas se conectan con neuronas en capas no necesariamente contiguas, saltando capas intermedias, y fueron propuestas en el 2015 [He y col., 2015a].

Cuando se diseñan redes neuronales profundas, se busca que obtengan un nivel alto de precisión sin que exista sobre-entrenamiento (overfitting). Sin embargo, se ha demostrado cómo en ocasiones, a medida que se añaden capas en una red, se alcanza un momento en el que la precisión obtenida, tanto en entrenamiento como en validación, comienza a disminuir. Como se ha explicado, durante el entrenamiento de una red, se calcula el gradiente del error obtenido con respecto a todos los pesos de la red. La forma de hacerlo es calcular primeramente el gradiente en la salida y propagarlo por las sucesivas capas hacia la entrada, de forma que su valor se va reduciendo. Cuando una red tiene muchas capas, el valor del gradiente propagado hasta las primeras capas tiene un valor muy pequeño, lo que se traduce en una actualización de los pesos demasiado lenta. Dicho de otra manera, las primeras capas de la red son las más lentas y difíciles de entrenar. Este problema se conoce como Desvanecimiento del Gradiente (Vanishing Gradient) [Hochreiter, 1998]. El paradigma ResNet permitió por primera vez entrenar redes muy profundas (de más de 100 capas); controlando con éxito éste problema.

Sus autores lanzaron tres arquitecturas inicialmente: ResNet50, ResNet101, ResNet152. El número que acompaña al nombre indica el número de capas de la red, que como se puede ver es mucho más elevado que en la arquitectura anterior. Sin embargo, a pesar de ello, esta red tiene una complejidad menor, es decir, el número de operaciones que realiza es menor. Esto es así porque implementa una forma de aprendizaje basada en minimización de residuos. Cada vez que la red incorpora una serie de capas que incluyen pesos se añade un atajo que vaya desde la entrada de la primera de las capas a la salida de la última, es decir, creando conexiones como la mostrada en la (Figura (2.21)).

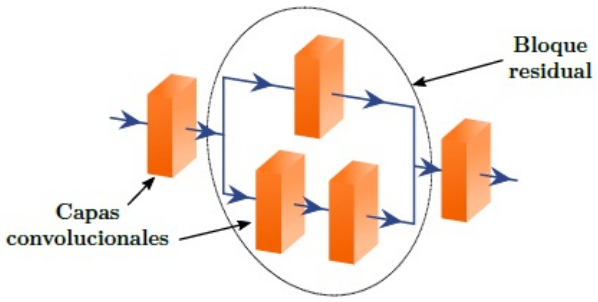


Figura 2.21: Ejemplo de bloque residual insertado.

La estructura formada por dos ramas paralelas afecta a la propagación del gradiente, haciendo que éste no disminuya en exceso y llegue a las primeras capas con un valor suficientemente alto, mejorando el entrenamiento de éstas.

- Bloques Identidad

Un Bloque Identidad es un bloque residual en el cual una de las dos ramas no realiza operación alguna. De esta forma, la salida del bloque coincide con la salida de la otra rama sumada con la entrada. Típicamente, un bloque residual (versión 2) se implementa como ilustra la (Figura (2.22)).

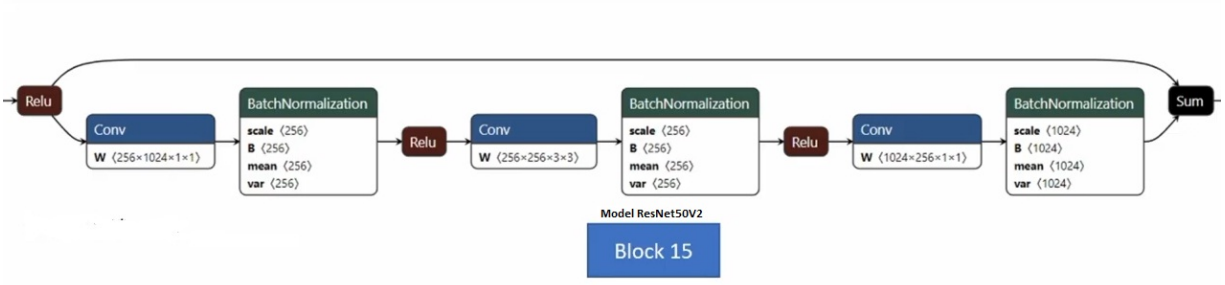


Figura 2.22: Ejemplo de bloque residual identidad.

Los datos que son "puenteados" no son modificados, el tensor que resulta de las etapas convolucionales es de las mismas dimensiones que los datos originales, por lo que no es necesario ajustar las dimensiones para sumarlos [Rivera, 2019].

- Bloques Convolucionales

Los denominados Bloques Convolucionales son otro tipo de bloques residuales con un comportamiento más parecido a la capa convolucional (Figura (2.23)). Aquí las dimensiones de x y $F(x)$ son distintas; ya sea porque se incrementó el número de canales o se cambiaron las dimensiones espaciales del tensor (o ambas). En tal caso, es necesario aplicar una operación a x para hacer congruentes las dimensiones.

Generalmente la primera capa convolucional usa un paso (stride) igual a dos (lo que reduce las dimensiones espaciales a la mitad, y la segunda capa convolucional cambia el número de canales. Por ello, para poder hacer la suma de la liga residual con los datos procesados, se agrega una capa convolucional (sin activación) en la liga puente con paso de dos y con el número de canales requeridos [Rivera, 2019].

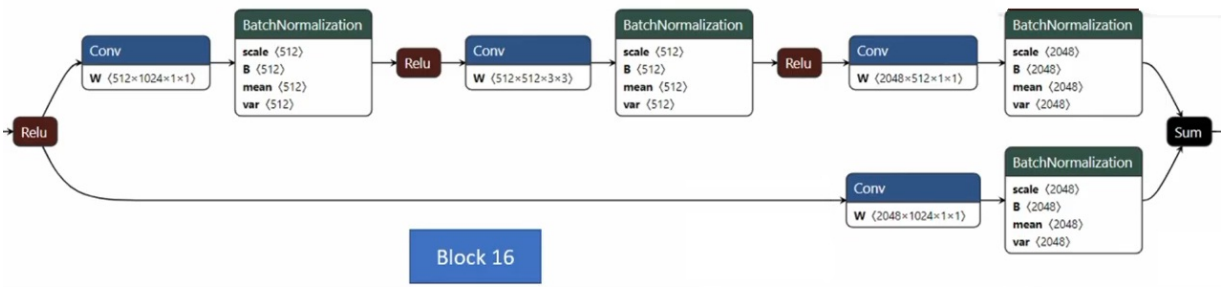


Figura 2.23: Ejemplo de bloque residual convolucional.

2.13. Métricas para calcular el desempeño de un modelo de red.

El objetivo del modelo, en el campo de la inteligencia artificial, es aprender patrones que generalizan bien cuando evaluamos en ellos datos que no han sido vistos durante el entrenamiento, en lugar de memorizar estos. Un número de métricas se utilizan para medir el desmpeño predictivo de un modelo.

Algunas de las métricas típicas son: exactitud (ACC), error del clasificador, precisión, exhaustividad o recuerdo, medición F1. Cada métrica mide un aspecto predictivo del modelo , así por ejemplo, la exactitud (ACC) mide la fracción de predicciones correctas, la precisión mide la fracción de positivos reales entre los ejemplos que se prevén como positivos, la recuperación mide cuantos positivos reales se predijeron como positivos y la medida F1 es la media armónica de la precisión y la recuperación.

Para el cálculo de estas métricas es importante definir las predicciones del modelo según:

• VP (verdaderos positivos) es la cantidad de positivos que fueron clasificados correctamente como positivos por el modelo.

• VN (verdaderos negativos) es la cantidad de negativos que fueron clasificados correctamente como negativos por el modelo.

• FN (falsos negativos) es la cantidad de positivos que fueron clasificados incorrectamente como negativos.

• FP (falsos positivos) es la cantidad de negativos que fueron clasificados incorrectamente como positivos.

Exactitud (Acurracy)

En general, ¿qué porcentaje de los datos clasifica correctamente?

$$\textbf{Exactitud} = \frac{VP + VN}{VP + VN + FN + FP}. \tag{2.10}$$

Error del clasificador

En general, ¿qué porcentaje de los datos clasifica incorrectamente? (1-Exactitud)

$$\textbf{Error} = \frac{FP + FN}{VP + VN + FN + FP}. \tag{2.11}$$

Precisión (Precision)

En general, fracción de las predicciones positivas que son correctas.

$$\textbf{Precisión} = \frac{VP}{VP + FP}. \tag{2.12}$$

Exhaustividad o Recuerdo (Recall)

En general, fracción de todas las instancias positivas que se clasifican correctamente como positivas.

$$\textbf{Recuerdo} = \frac{VP}{VP+FN}. \tag{2.13}$$

Medida F1 (F1 mesure)

Combina precisión y recuerdo

$$\textbf{MedidaF1} = 2 * \frac{Precisión * Recuerdo}{Precisión + Recuerdo} = \frac{2 * VP}{2 * VP + FN + FP}. \tag{2.14}$$

Es importante recordar que en la clasificación multiclase la respuesta predicha es la clase con la puntuación máxima predicha, a diferencia del proceso de los problemas de clasificación binaria dónde se elige un umbral de puntuación para realizar las predicciones.

2.14. Matriz de confusión.

En el campo de la inteligencia artificial la matriz de confusión es una herramienta que permite la visualización del desempeño del algoritmo que se emplea en el aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real (como fueron predichas en relación a la clase a la que pertenecían). Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

La matriz tiene la siguiente estructura. (Figura (2.24))

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 2.24: Matriz de confusión.

2.15. Métodos para la visualización e interpretación de un modelo de red entrenado.

Existen muchos métodos para la visualización e interpretación del comportamiento de las redes neuronales convolucionales. En general, los podemos agrupar en tres categorías princi-

pales [Hernández Hernández, 2020]:

• Métodos basados en gradientes

Como su nombre indica, estos métodos se basan en hacer una propagación hacia atrás del gradiente de una clase con respecto a la imagen de entrada para resaltar aquellas partes de la misma que más influyen en la predicción de la red. La forma más básica de estos métodos es la que se describe en [Ron Kohavi, 1997] con el cálculo directo del gradiente.

• Métodos basados en perturbaciones

Este tipo de métodos funcionan perturbando, de alguna manera, la imagen de entrada y viendo qué efecto tiene dicha perturbación en la probabilidad de salida de la red para una determinada clase.

Implican perturbar la intensidad de los píxeles de la imagen de entrada a la red neuronal con un ruido mínimo y observar el campo de probabilidad que tiene la predicción de acuerdo al ruido. La idea detrás de esto es que los píxeles que contribuyan en mayor medida a la clasificación acertada del objeto a clasificar, una vez alterado su valor, reducirán el resultado de la clasificación, permitiendo así identificar dichos píxeles mediante mapas de calor [M. Ancona y Gross, 2017].

Uno de estos métodos conocido como LIME [M. Ribeiro y Guestrin, 2016] (por sus siglas en inglés) basa su funcionamiento en agrupar píxeles individuales en grupos con valores similares llamados superpíxeles. Una vez agrupados, se comienza a activar y desactivar los grupos, generando imágenes de manera iterativa, calculando predicciones y ponderando la importancia de los grupos para la predicción. Esto permite visualizar qué características toma en cuenta el modelo para realizar la predicción.

En la (Figura (2.25)) se muestra el resultado de implementar el método LIME con 1000 iteraciones, utilizando la arquitectura Xception con la especie Purshia plicata [Ribeiro y col., 2016], resaltando en la imagen las características mayormente utilizadas por el modelo para realizar la predicción.

Figura 2.25: Visualización de las características que toma en cuenta el modelo para realizar la predicción según LIME.

• Métodos basados en CAM (Class Activation Mapping)

La salida de estos métodos es, habitualmente, un mapa de calor que se obtiene como la combinación lineal ponderada de mapas de activación de capas de convolución, normalmente, de las más profundas. Lo que diferencia a unos métodos de otros, dentro de esta categoría, es la forma de calcular los pesos en dicha combinación lineal. Los pesos de dicha combinación vienen dados por su relevancia de cara a la predicción final. Esta relevancia la conocemos gracias a la arquitectura de la que parte y es dependiente el método (Figura (2.26)).

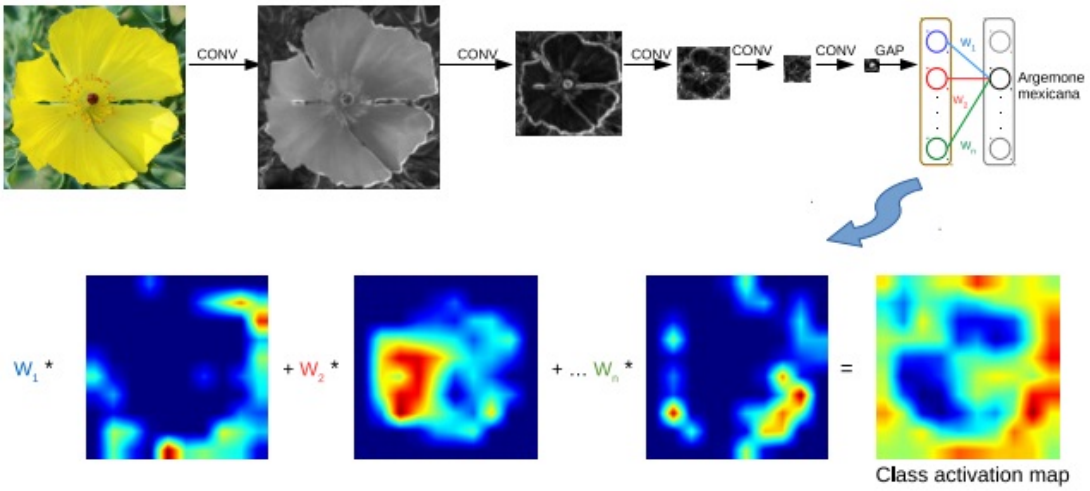


Figura 2.26: Metodología de generación de la imagen CAM.

• Método Grad-CAM (Gradient-weighted Class Activation Mapping)

El método Grad-CAM o mapeo de activación de clases ponderado por gradientes (Gradient-weighted Class Activation Mapping) [Selvaraju y col., 2019] es una generalización del método CAM y propone una técnica para producir 'explicaciones visuales' para decisiones de una gran clase de modelos basados en redes neuronales convolucionales (CNN), haciéndolos más transparentes y explicables.

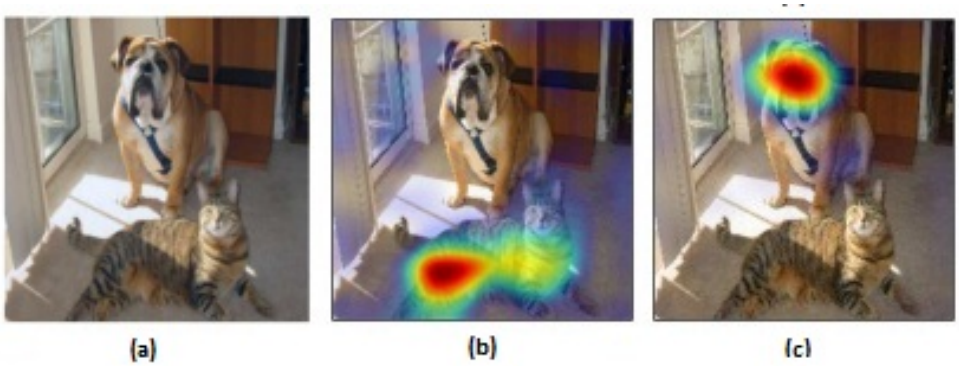


Figura 2.27: (a)Imagen original con un gato y un perro. (b, c) Imágenes Grad-CAM que localiza regiones discriminatorias de clase. Las regiones rojas corresponden a la puntuación más alta para la clase.

Este método utiliza los gradientes de cualquier concepto de destino (por ejemplo, 'perro' en una red de clasificación o una secuencia de palabras en una red de subtítulos) que fluye

hacia la capa convolucional final para producir un mapa de localización que destaca las regiones importantes en la imagen para predecir el concepto (Figura (2.27)) [Selvaraju y col., 2019].

Las imágenes de Grad-CAM ayudan a establecer una confianza adecuada en las predicciones de las redes profundas y permiten discernir con éxito una red profunda más fuerte de una más débil incluso cuando ambas hacen predicciones idénticas en función de la valoración del área en la imagen que utiliza la red para la predicción que realiza.

Capítulo 3

Estado del Arte

3.1. Aprendizaje por Transferencia

Hay una condición muy deseada y explotada en las redes neuronales y es su capacidad de transferir el conocimiento adquirido en un dominio a otro donde los datos son generalmente escasos [Neyshabur y col., 2021]. El aprendizaje por transferencia consiste entonces en tomar un modelo de red previamente entrenado que ha aprendido a resolver un problema y utilizar esos pesos pre-entrenados como inicialización del modelo para su entrenamiento en el dominio de datos del nuevo destino. Con este proceder se espera que la red sea capaz de hacer uso de una reutilización de características generales ya aprendidas cuando se entrena en los nuevos datos.

La transferencia de aprendizaje y la adaptación de dominio se refieren a la situación en la que lo que se ha aprendido en un entorno, se explota para mejorar la generalización en otro entorno [Goodfellow y col., 2016].

El aprendizaje por transferencia tiene ventajas significativas si no hay datos suficientes para entrenar un modelo. En otras palabras, el aprendizaje por transferencia nos permite entrenar nuestro modelo de aprendizaje profundo con un número de datos relativamente pequeño y encontrar soluciones favorables a nuestro problema si las características del modelo aprendidas en la primera tarea son generales o similares a la tarea que realizamos. Para el primer caso, la transferencia de características generales, que será lo que utilizaremos en este trabajo es de suma importancia que el entrenamiento inicial se realice en una base de datos muy amplia.

En correspondencia con estudios previos que afirman que las características que identifican las redes neuronales convolucionales se vuelven más especializadas a medida que se avanza hacia las capas superiores del modelo, es de esperar entonces que la reutilización de características en la red esté ocurriendo en capas que están más cerca de la entrada dado que son estas capas la encargadas de manejar características que podemos considerar similares para diferentes dominios de datos [Stanford, 2021].

Los módulos en las capas inferiores, capas cercanas a la entrada, están a cargo de las características generales y los módulos en las capas superiores son más sensibles a la pertur-

bación de sus parámetros, lo cual quiere decir que los valores de los pesos se encuentran muy cercanos unos de otros y son muy sensibles a un incremento de esa diferencia [Neyshabur y col., 2021]. Para arquitecturas en capas como VGG, cada capa es un módulo. Para ResNets, un módulo puede ser un bloque residual, que contiene varias capas y una conexión de salto.

Entre los escenarios principales del aprendizaje por transferencia tenemos [Stanford, 2021]:

• **Como extractor de funciones fijas.** Consiste en tomar un modelo CNN previamente entrenado en la base de datos de partida (en nuestro caso ImageNet) y eliminar las capas del clasificador del modelo (cabeza del modelo, las salidas de esta capa son las puntuaciones de 1000 clases para una tarea como ImageNet), luego utilizar el resto del modelo como un extractor de características fijas para el entrenamiento de un nuevo clasificador y una capa densa en correspondencia con el número de clases a definir en el nuevo conjunto de datos con función de activación del mismo nombre.

• **Ajuste Fino (Fine-tuning).** La segunda estrategia es no solo reemplazar y volver a capacitar al clasificador en la parte superior del modelo en el nuevo conjunto de datos, sino también ajustar los pesos de la red preentrenada al permitir la actualización de sus pesos mediante la propagación hacia atrás del error. Es posible ajustar todas las capas de la red, o es posible mantener algunas de las capas anteriores fijas (debido a problemas de sobreajuste) y solo ajustar una parte del nivel superior de la red. Esto está motivado, como se menciona anteriormente, por la observación de que las capas bajas contienen características más genéricas (por ejemplo, detectores de bordes o detectores de manchas de color) que deberían ser útiles para muchas tareas, pero las capas superiores se vuelven progresivamente más específicas para los detalles de las clases

En general, no es obvio que habrá un beneficio al utilizar el aprendizaje por transferencia en el dominio hasta después de que se haya desarrollado y evaluado el modelo, pudiera necesitarse modificar algún hiperparámetro (por ejemplo función de pérdida o el optimizador usado, todo en relación al problema especifico que buscamos solucionar). Pero se pueden describir tres posibles beneficios al utilizar el aprendizaje por transferencia [Torrey y Shavlik, 2010]: (Fig. (3.1))

-Comienzo superior. La habilidad inicial (antes de refinar el modelo) en el modelo de origen es mayor de lo que sería de otra manera (se comienza con un cierto conocimiento inicial de la tarea ,es decir hay una favorable inicialización de las capas del modelo).

-Pendiente más alta. La tasa de mejora de la habilidad durante el entrenamiento del modelo fuente es más pronunciada de lo que sería de otra manera (se logra un entrenamiento más rápido).

-Asíntota superior. La habilidad de convergencia del modelo entrenado es mejor de lo que sería de otra manera (se alcanza un mejor valor final de desempeño del modelo).

La reutilización de parámetros, consecuencia de que los dominios de datos comparten características visuales similares, como se dijo anteriormente, juega un papel importante en el aprendizaje por transferencia . Pero incluso para los dominios de destino más distantes se pueden observar mejoras en el rendimiento a partir del aprendizaje por transferencia lográndose, por lo general, que el entrenamiento converja mucho más rápido en todos los

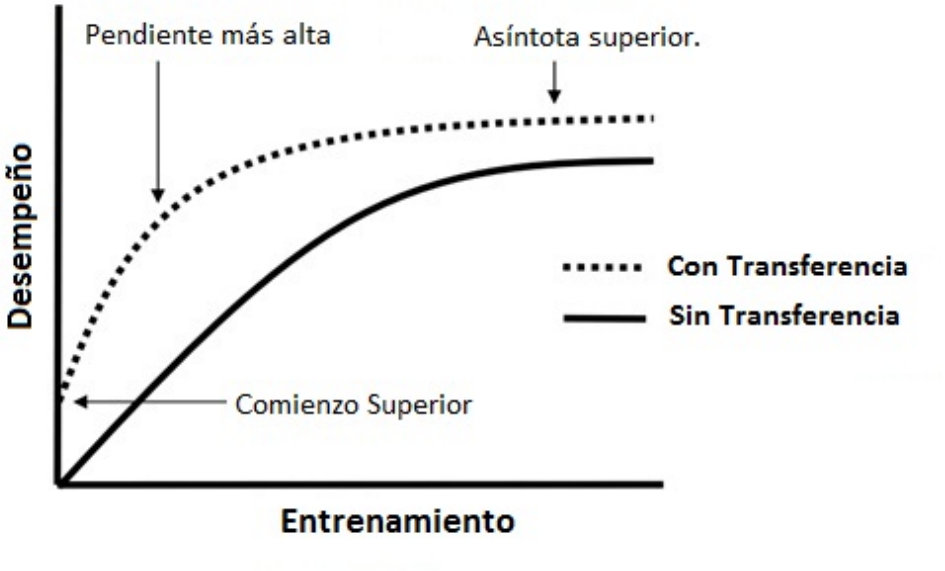


Figura 3.1: Tres formas en las que la transferencia podría mejorar el aprendizaje.

casos [Neyshabur y col., 2021]. Esto sugiere beneficios adicionales de los pesos previamente entrenados que no solo provienen directamente de la reutilización de funciones.

3.2. Aprendizaje Curricular

Las experiencias de los métodos cognoscitivos en la enseñanza en humanos y animales nos muestran resultados favorables cuando buscamos transmitir el conocimiento a partir de trabajar con ejemplos prácticos que no se enseñan al azar sino que se organizan en un orden significativo. Este orden hace que gradualmente se ilustren más conceptos y a su vez se les haga aumentar en complejidad. ¿Cómo relacionar los conocimientos que nos deja la ciencia cognitiva y su interacción con el aprendizaje automático que se vieron abordados inicialmente en [Elman, 1993], donde los autores se realizaron la siguiente pregunta: ¿Pueden los algoritmos de aprendizaje automático beneficiarse de una estrategia de capacitación similar? La idea básica es comenzar de a poco, aprender aspectos más fáciles de la tarea o subtareas más fáciles, y luego aumentar gradualmente el nivel de dificultad.

En [Bengio y col., 2009] se define la presencia de un currículo en el entrenamiento cuando la entropía de las distribuciones aumenta. Esta distribución, en imágenes, podemos referirla a como las ordenamos para hacer que la complejidad visual de las mismas aumente. Se puede pensar entonces en una secuencia de conjuntos de entrenamiento donde comenzamos con un pequeño conjunto de ejemplos fáciles y terminamos con el conjunto de entrenamiento final. Al asegurar que la entropía aumente, aseguramos que aumente la diversidad en los ejemplos de entrenamiento. De esta forma los pesos de ejemplos particulares se agregan al conjunto de entrenamiento a medida que este crece.

Explorar diferentes estrategias curriculares necesita mucho trabajo, algunas pueden ser

muy específicas para tareas particulares así como definir qué son ejemplos fáciles o, de manera equivalente, ejemplos que clasifican en una secuencia que ilustra los conceptos más simples. Sin embargo, definir ejemplos menos 'ruidosos' es una manera simple de ayudarse [Bengio y col., 2009].

Los resultados experimentales aportan evidencia para afirmar que el entrenamiento curricular mejora el aprendizaje de los algoritmos automáticos y que se puedan obtener mejores resultados en conjuntos de datos con un plan de estudios de estrategias apropiado. Una estrategia de currículum apropiada actúa para ayudar al proceso de capacitación (convergencia más rápida hacia mejores soluciones), e incluso se regulariza, dando lugar a un menor error de generalización para el mismo error de capacitación [Bengio y col., 2009].

Las estrategias curriculares también están relacionadas con el aprendizaje de transferencia (o multitarea) y el aprendizaje permanente [Thrun, 1996]. Las estrategias de aprendizaje del currículo pueden verse como una forma especial de transferencia de aprendizaje donde las tareas iniciales se utilizan para guiar hacia un desempeño mejor en la tarea final.

Podemos concluir que el aprendizaje curricular agrega la noción de guiar el proceso de optimización, ya sea para converger más rápido o, hacia mejores mínimos locales

3.3. Curriculum por textura

Las imágenes usadas en un entrenamiento, a menos que características específicas de las mismas la limiten contienen patrones de altas y bajas frecuencias. Las características que podemos obtener de la imagen observando su espectro de frecuencias se resumen en [Gonzalez y Wintz, 1996]:

- Zonas homogéneas en la imagen darán lugar a que la energía del espectro esté concentrada mayoritariamente en las bajas frecuencias.

- Zonas con muchos bordes, transiciones frecuentes de los nıveles de color darán lugar a un espectro con componentes de alta frecuencia.

- Si en la imagen existe alguna regularidad (patrones que se repiten) darán lugar a picos de igual intensidad separados una distancia regular.

Como se ha referido con anterioridad, es sabido que las CNN pueden clasificar imágenes de manera efectiva y para esto realizan convoluciones en una imagen utilizando núcleos que modifican sus valores (aprenden) durante el proceso de entrenamiento. estas operaciones de convolución provocan un fuerte sesgo inductivo hacia la equivalencia espacial local. El fuerte sesgo inductivo les permite aprender pequeñas transformaciones espaciales y características locales en una imagen. Junto con el sesgo inductivo espacial, las CNN también están sesgadas hacia la información de alta frecuencia, o información de textura en las imágenes [Geirhos y col., 2018]. Este sesgo limita la capacidad de una CNN de utilizar información de baja frecuencia, o de forma, la cual puede ser útil para la clasificación. [Sinha y col., 2021]

En un esquema de aprendizaje curricular basado en textura, la información que caracteriza en detalle a la imagen, información de alta frecuencia, se dispone para la red en un proceso

progresivo a medida que avanza la capacitación. Por lo tanto, al principio del entrenamiento, se restringe la información de textura disponible y se obliga a la red a optimizar el objetivo del entrenamiento usando la información de baja frecuencia presente en la entrada. Al controlar la información de textura de esta manera, se alienta activamente a las CNN a predecir la salida (por ejemplo, la etiqueta de clase) utilizando la información de baja frecuencia disponible durante la capacitación.

Capítulo 4

Metodología

4.1. Aprendizaje por Transferencia para los modelos de estudio

En el presente trabajo se explotan las ventajas descritas para el aprendizaje de transferencia buscando obtener una inicialización de los pesos lo más favorable posible, para los modelos usados, antes de comenzar el aprendizaje curricular. Para ello la capa densa o clasificador que se va a usar se establece para la cantidad de clases a definir en nuestra base de datos y es entrenado después de inicializar los modelos en los pesos de su entrenamiento en la base de datos de ImageNet. Luego se sigue un esquema de ajuste fino para todas las capas. (Figura. (4.1))

4.2. Implementación del curriculum por textura en los modelos

En la investigación que desarrolla este trabajo para modificar el sesgo inductivo, propio de redes neuronales convolucionales, se cambia directamente la arquitectura de la red, con la inclusión en la misma de capas de filtro gaussiano y a continuación del aprendizaje de transferencia y ajuste fino descrito en la base de datos ImageNet, se entrenan los modelos en el conjunto de datos propio siguiendo un entrenamiento curricular basado en textura. El uso de la capa de convolución separable en su primera etapa, depthwise convolution, permite operar con los filtros gaussiano a la salida de cada canal de una capa convolucional clásica (Figura (4.2)).

Los núcleos gaussianos incluidos no están entrenados mediante el descenso de gradiente es decir no requieren actualizaciones de gradiente; por lo tanto, no estamos agregando parámetros entrenables adicionales a la red.

El hiperparámetro σ de los núcleos gaussianos, como se explicó anteriormente, controla la cantidad de la salida que se "desenfocará"después de una operación de convolución, ya que al

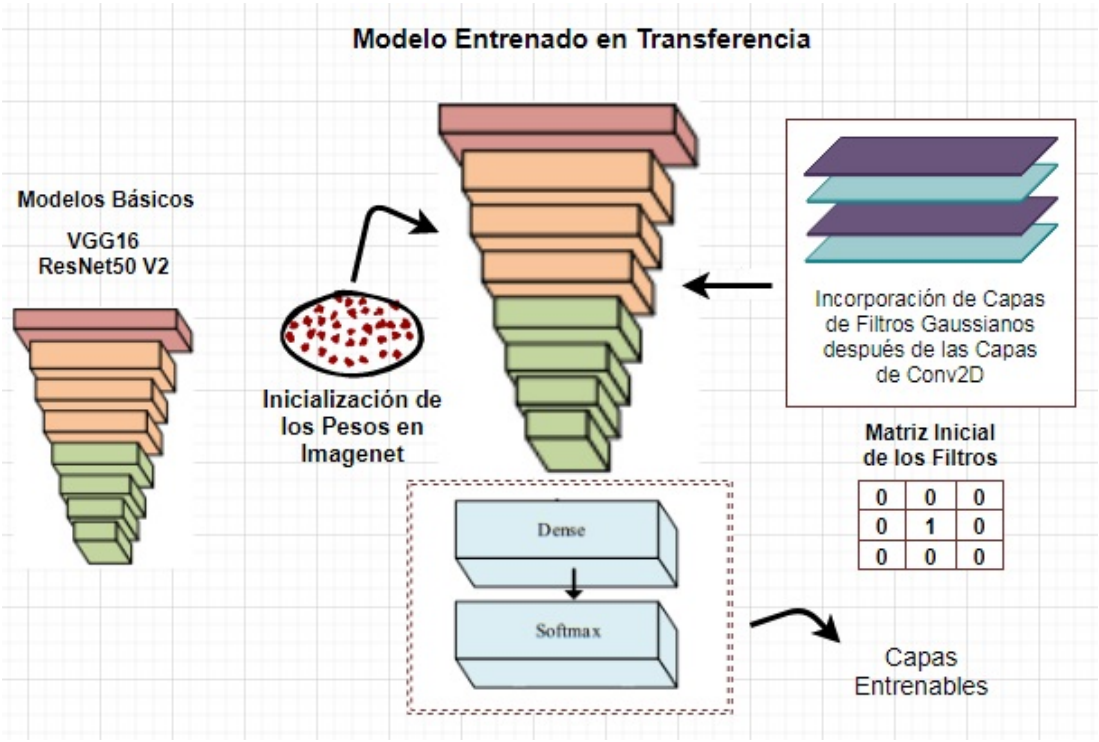


Figura 4.1: Esquema del entrenamiento de transferencia usado.
Los modelos clásicos son modificados con la inclusión de la capas de filtro que se usarán en el entrenamiento curricular, pero debido al valor inicial fijado en la matriz de los filtros durante el entrenamiento por transferencia y posterior ajuste fino de todas las capas del modelo, estos no originan diferencias a la salidas de las capas convolucionales.

aumentar σ se produce una mayor cantidad de suavizado. En el entrenamiento por suavizado, como también se le nombra a este entrenamiento curricular, el valor de este hiperparámetro se maneja escalonadamente en sentido decreciente para lograr, como se explicaba, que la información de alta frecuencia se ofrezca a la red en un proceso progresivo.

Al aplicar el desenfoque gaussiano a la salida de las capas convolucionales de los modelos clásicos, suavizamos las características de las salidas y reducimos la información de textura que la capa convolucional puede propagar hacia la capa que le precede y así se garantiza que se propague una información textural mínima a través de la red. Si las CNN no tuvieran un fuerte sesgo hacia la textura, el método de entrenamiento obstaculizaría significativamente la capacidad de la red para aprender sobre la textura en las imágenes pero dado que la textura se prefiere sobre la forma, la red puede adaptarse a una mayor cantidad de contexto de textura a medida que avanza el entrenamiento y se varía el valor de σ.

Al reducir el ruido y los artefactos de solapamiento (aliasing), mediante los filtros gaussianos, en los mapas de características que originan las operaciones de convolución, se asegura que estos no estén sesgados por el ruido en las inicializaciones de los pesos, ya que se ha demostrado que tal sesgo al principio del entrenamiento es crítico para el aprendizaje. [Jastrzebski y col., 2020]

En el entrenamiento curricular por suavizado que usamos para entrenar los modelos

Model: ResNet50V2

Layer (type)	Output Shape	Param #
input_37 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
depthwise_conv2d_377 (Depthw	(None, 224, 224, 64)	576
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
depthwise_conv2d_378 (Depthw	(None, 224, 224, 64)	576
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
depthwise_conv2d_379 (Depthw	(None, 112, 112, 128)	1152

Figura 4.2: Sección del modelo ResNet50V2 con capas Depthwise incluídas después de cada capa de convolución 2D.

estudiados, se establece un plan curricular para el valor de sigma de los filtros gaussianos que fija los valores de σ en (1, 0.8, 0.6, 0.4, 0.2) es decir inicializamos en σ=1 y vamos decreciendo 0.2 hasta alcanzar el valor σ=0.2. Para cada valor fijado a σ se entrena el modelo en 50 épocas, para después cambiar el valor del parámetro de la desviación estándar del filtro al valor que le precede. Cuando se reinicia el entrenamiento con el nuevo valor del parámetro se establece como inicialización de los pesos, para el modelo, el mejor valor alcanzado en el grupo de entrenamientos que le precedieron (Figura (4.3)). Para ello usamos la técnica de la API de Keras conocida como "parada anticipada"(early-stopping), método que permite especificar una gran cantidad arbitraria de épocas de entrenamiento (en nuestro caso 50) y detener el entrenamiento una vez que el rendimiento del modelo deja de mejorar para el conjunto de datos de validación [“Keras API reference / Callbacks API.” 2021]. Éstas paradas tempranas evita el sobreajuste de los modelos durante el aprendizaje.

El método de parada temprana permite monitorear varias métricas durante el entrenamiento del modelo, en nuestro caso seguimos el comportamiento de la pérdida en el conjunto de datos de validación (está disponible a través del nombre val−loss en Keras). De esta forma el entrenamiento se detiene cuando la medida de desempeño elegida deje de mejorar (se alcance un valor mínimo durante las épocas de entrenamiento). A menudo, la primera señal de que no hay más mejoras puede que no sea el mejor momento para dejar de entrenar. Esto se debe a que el modelo puede deslizarse hacia una meseta sin mejora o incluso empeorar un poco antes de mejorar mucho [Brownlee, 2020]. Para superar este problema se permite agregar un retraso al disparador en términos de la cantidad de épocas en las que no nos gustaría ver

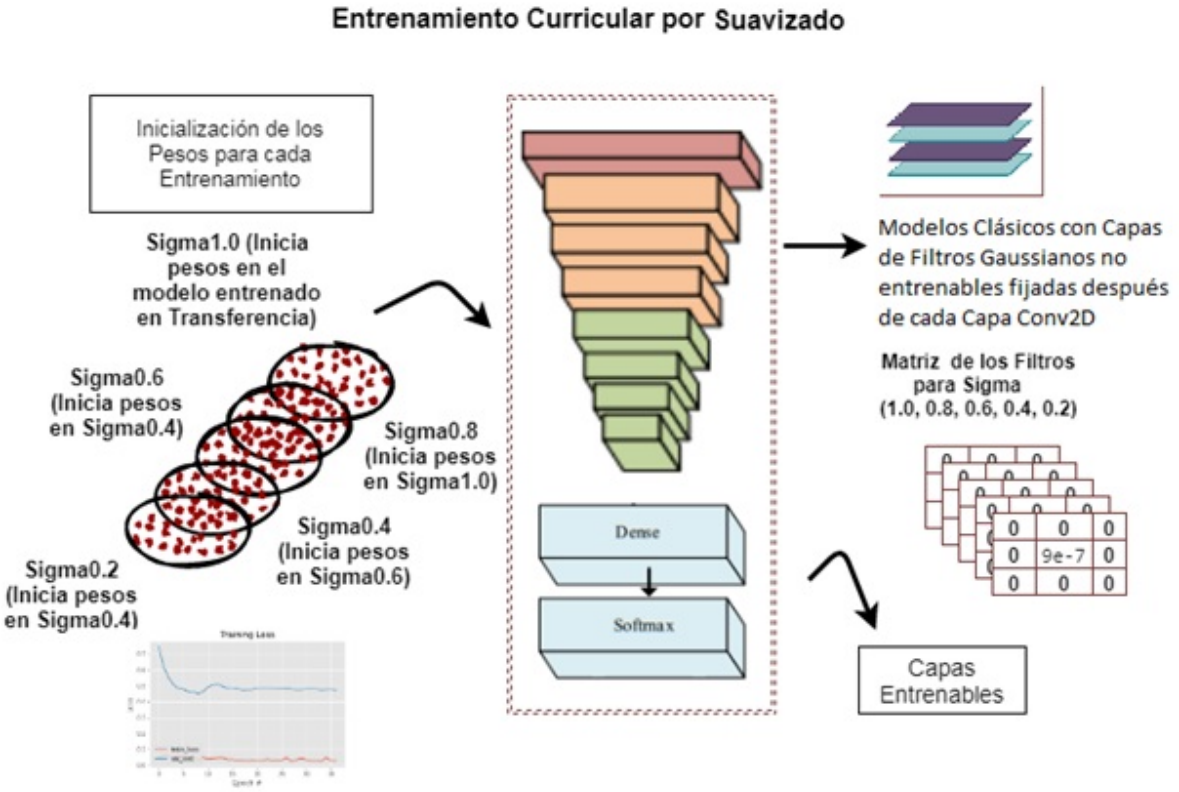


Figura 4.3: Esquema del entrenamiento curricular por suavizado usado.
Cada entrenamiento sucesivo conlleva la inicialización del modelo en los pesos resultados del entrenamiento previo y el establecimiento de los valores de la matriz de los filtros gaussianos en correspondencia al valor de sigma fijado para ese entrenamiento. El valor inicial de los pesos para σ=1 es tomado del entrenamiento de transferencia con inicialización del modelo clásico en los pesos de ImageNet.

ninguna mejora, el argumento se nombra paciencia (patience). En nuestro estudio ésta se fijó en 12 épocas. Además se requiere una devolución de llamada adicional que guardará el mejor modelo observado durante el entrenamiento para su uso posterior, esta se nombra ModelCheckpoint y la usaremos para guardar los pesos del mejor valor alcanzado en el entrenamiento para la medida de rendimiento elegida en el conjunto de datos de validación (val−loss para nuestro caso) [“Keras API reference / Callbacks API.” 2021]. De esta forma, como se dijo anteriormente, al pasar de un valor de σ a otro, en los entrenamientos sucesivos, se asegura que se inicie el aprendizaje en las mejores condiciones del entrenamiento que le precedió para el valor de σ anterior.

Capítulo 5

Características de la base de datos de imágenes de plantas de albahaca usada en los entrenamientos.

Para la diferenciación del desarrollo de las plantas se establecieron cuatro tratamientos en base a diferentes niveles de Nitrógeno (N), el cual se suministró mediante la aplicación de nitrato mezclado en el agua de riego, este aporte se realizó durante todo el experimento con cada riego desde el momento del trasplante. Los niveles fueron: 4, 8, 12 y 16 mEq / L y denominados para los propósitos de esta investigación como nivel I, II, III y IV respectivamente. Los micronutrientes se suministraron utilizando el producto comercial Ultrasol micro Rexene BSP Mix en una dosis de 80 g / m3 con el fin de asegurar que los cambios físicos mostrados por las plantas se debieran únicamente a cambios en las dosis de N y no a la falta de cualquier otro nutritivo.

Durante el ciclo de crecimiento de las plantas de albahaca no se observaron problemas de plagas o enfermedades, ni ningún tipo de estrés por factores abióticos que pudieran ocasionar desviaciones en el desarrollo de las plantas así como también se mantuvo controlada la temperatura promedio registrada dentro del invernadero y la humedad relativa promedio.

Se analizaron un total de 48 plantas, 12 plantas por tratamiento (Figura (5.1)). Para cada planta, se capturaron un total de 18 imágenes desde diferentes puntos de vista para lo cual la planta se colocó sobre un plato giratorio y se rotó cada 20 grados mientras la cámara permanecía fija enfocando la planta sobre un fondo blanco, éste trabajo permitió que las imágenes tomadas de la misma planta alcancen a tener un aspecto muy diferente debido a las características morfológicas de la albahaca (Figura (5.2)).

De ésta forma el conjunto de datos consta de un total de 864 imágenes (con tamaño de 224×224 píxeles) obtenidas de 48 plantas, de las cuales 32 plantas (8 plantas por tratamiento) se consideraron para entrenamiento y 16 plantas para prueba (4 plantas por tratamiento). Por lo tanto, hay 576 imágenes (144 por tratamiento) en el conjunto de entrenamiento y 288 imágenes en el conjunto de prueba (72 por tratamiento). Todas las imágenes de una planta están solo en uno de los conjuntos. Para un mejor análisis del conjunto de datos, se crearon

Figura 5.1: Ejemplos de las imágenes. Nivel I (columna 1), nivel II (columna 2), nivel III (columna 3) y nivel IV (columna 4).

Figura 5.2: Imágenes de la misma planta tomadas con 6 ángulos de rotación diferentes.

3 particiones distintas de los datos, donde las plantas consideradas para los conjuntos de prueba son todas diferentes.

Para eliminar completamente la influencia del fondo y la base, visible en las imágenes, en el entrenamiento de los modelos, realizamos también el entrenamiento sobre dos modificaciones realizadas a los datos que dejan a la plantas sobre un fondo negro. De esta manera podemos validar también los valores fijados a los parámetros de los entrenamientos y las consideraciones tomadas a partir de las imágenes GradCAM.

En la primera de las transformaciones las imágenes sin procesar se recortaron automáticamente para contener solo la planta. Cada una de las imágenes RGB se transformó al espacio de color HSV (tono, saturación y valor) usando la biblioteca OpenCV [OpenCV] para detectar

aquellas regiones de píxeles que pertenecen a la planta especificando valores máximos y mínimos para cada canal. Una vez binarizada la imagen, se procesó la imagen para eliminar el ruido. Finalmente, se determinó un cuadro delimitador de toda la región de la planta y se ajustó para extraer un parche cuadrado con fondo negro. En la segunda de las transformaciones se adicionó a este trabajo una segmentación manual de las plantas en las imágenes con precisión a nivel de píxel (Figura (5.3)).

Figura 5.3: Ejemplos de las imágenes sobre fondo negro con segmentación automática. Las imágenes mantienen un tamaño de 224×224 píxeles.

Capítulo 6

Resultados.

6.1. Entrenamiento de los modelos de redes del estudio.

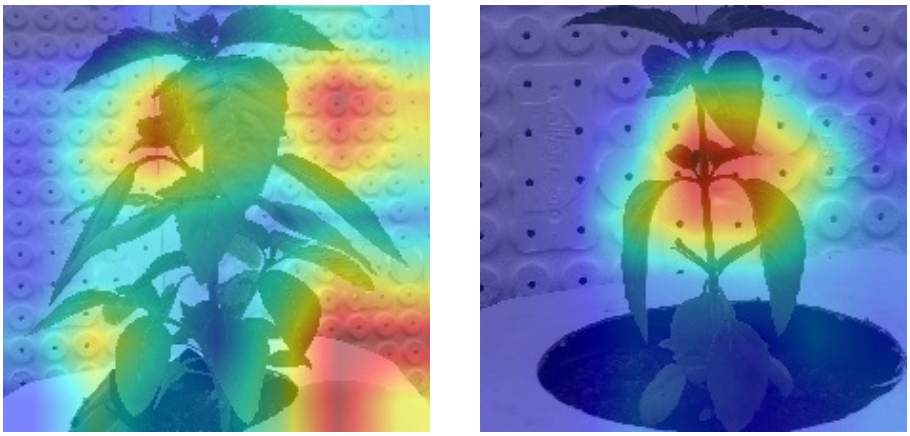
En la investigación realizada se trabajó primeramente en la optimización de los hiperparámetros en el entrenamiento clásico (Transferencia y Ajuste Fino) de los modelos.

6.2. Evaluación del modelo entrenado mediante la valoración de las imágenes de localización basada en gradientes, imágenes GradCAM

La evaluación visual de la posición y tamaño del área de atención que usaba el modelo entrenado para la clasificación de las imágenes de validación, obtenidas por medio del la localización basada en gradientes, ayudó en la selección de los valores óptimos de los parámetros del entrenamiento para minimizar la influencia del fondo en el etiquetado de las imágenes (Figura (6.1)).

En el desarrollo de esta investigación, de manera general en las imágenes analizadas, el entrenamiento curricular termina con áreas GradCam más pequeñas y mejor ubicadas sobre la zona de interés de la imagen clasificada. Dicha reducción del área de activación visible, parece relacionarse con un aumento del valor de clasificación de la imagen como perteneciente a una clase específica. (Figura (6.2))

El uso gradual de los filtros y el continuo reentrenamiento permite al modelo mejorar la atención sobre la planta pudiéndose apreciar, en las imágenes Gradcam, que las zonas de activación presentan poca área sobre el fondo de la imagen incluso en aquellas imágenes que clasifica erróneamente.



(a) Selección incorrecta **(b)** Selección correcta

Figura 6.1: Visualización de áreas de clasificación en modelos entrenados haciendo uso del método GradCam.

6.3. Valores fijados para los hiperparámetros de los entrenamientos

Tras sucesivos entrenamientos y como resultado de la comparación de las métricas obtenidas en estos se pudieron fijar los valores óptimos ha asignar a los hiperparámetros de los entrenamientos. El uso de la información visual que aportaba las imágenes GradCam obtenidas de los modelos entrenados aseguraba, como se dijo anteriormente, la adecuada ubicación de la zona de atención de la red sobre el área que ocupa la planta en las diferentes imágenes de validación y de esta forma no solo se consideraba el criterio numérico obtenido en las métricas del entrenamiento sino que además se evaluaba visualmente que estas correspondían a un adecuado enfoque de la red sobre el objeto de interés en la imagen, la planta.

Los hiperparámetros cuya influencia fue analizada en los resultados obtenidos en los entrenamientos de los modelos fueron:

• Valor de la tasa de aprendizaje (Learning rate) utilizada en los dos momentos del entrenamiento. (se fijó $Lr = 1e^{-3}$ durante entrenamiento de transferencia y $Lr = 1e^{-5}$ para el ajuste fino)

• Selección de las transformaciones a usar en el ImageDataGenerator así como la comprobación de los valores y rangos adecuados a fijar en ellas. Se decidió establecer:

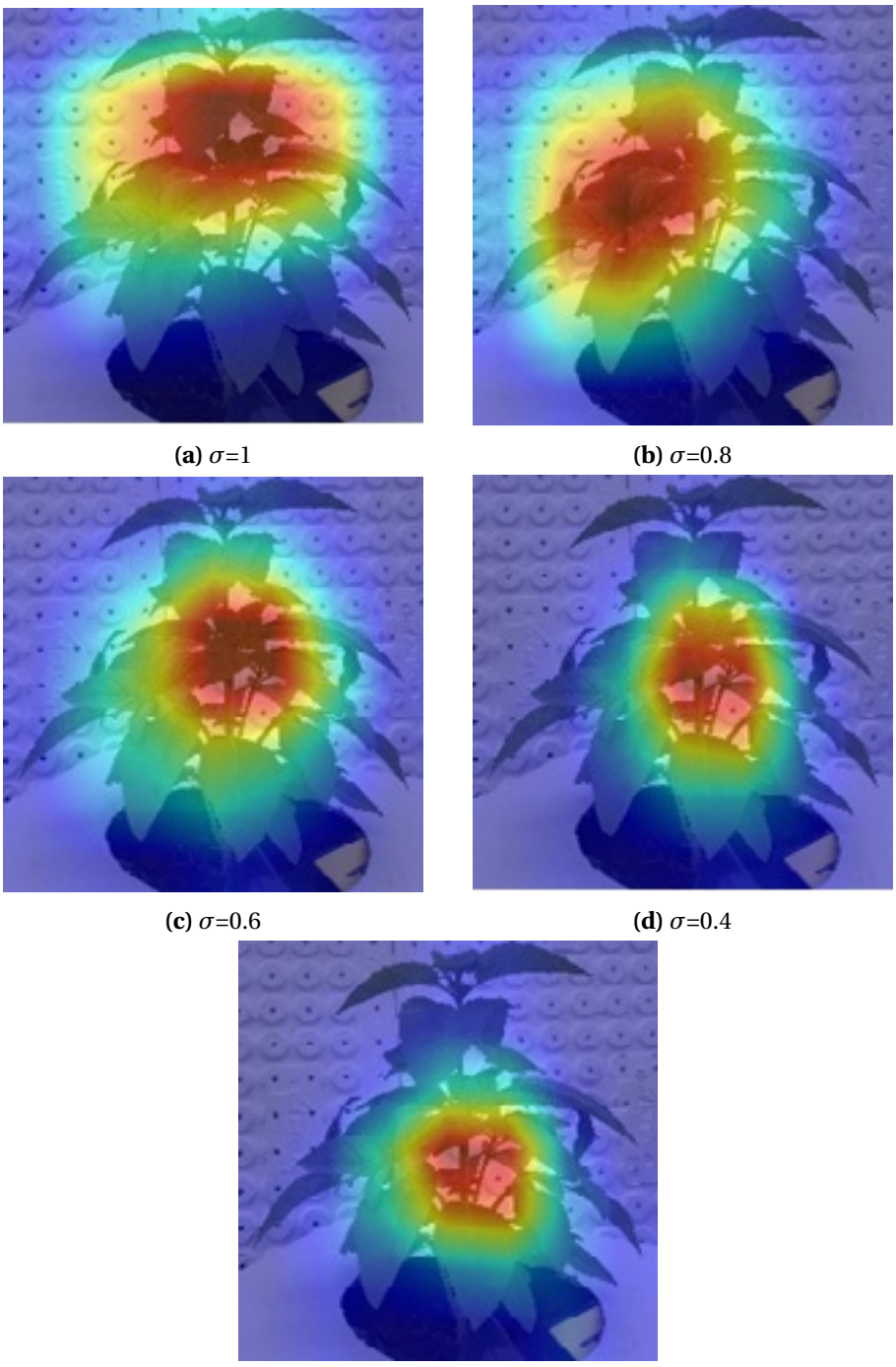
✓Rotation.range: 30 grados

✓Zoom.range: [0.8, 1.1]

✓Horizontal.flip: True

✓Brightness.flip: [0.8, 1.1]

✓Fill.mode: constant, cval: 0.0



(a) σ=1 **(b)** σ=0.8

(c) σ=0.6 **(d)** σ=0.4

(e) σ=0.2

Figura 6.2: Visualización del área Gradcam para una misma imagen al ser clasificada por el modelo entrenado en los diferentes valores de σ.

6.4. Entrenamiento clásico.

Con las condiciones anteriormente fijadas se obtuvieron las métricas de los entrenamientos de los dos modelos de estudio, VGG16 y ResNet50V2, en los tres splits de la base de datos para

establecerlas como punto de comparación contra el esquema de entrenamiento curricular propuesto. Se incluye además el entrenamiento de los modelos con la imagen de la planta sobre un fondo negro. Los resultados obtenidos se aprecian en la Tabla (6.1).

Tabla 6.1: Resultados del entrenamiento clásico en los dos modelos del estudio.

Transferencia de Aprendizaje y Ajuste Fino. Optimizador SGD		
Imágenes de la planta sobre el fondo original		
Accuracy (%):		
	VGG16	ResNet50V2
Split1	85.532 ± 0.818	85.185 ± 2.234
Split2	78.704 ± 1.488	73.495 ± 2.164
Split3	86.343 ± 0.818	81.019 ± 0.989
Promedio	83.526 ± 1.041	79.900 ± 1.796
Imágenes con segmentación manual de la planta sobre un fondo negro		
Split1	74.884 ± 2.526	82.176 ± 1.380
Split2	69.676 ± 0.600	70.370 ± 2.526
Split3	94.560 ± 0.989	85.532 ± 0.989
Promedio	79.707 ± 1.372	79.359 ± 1.632

Los valores obtenidos en el entrenamiento clásico en los modelos de estudio trae varios resultados que son de interés analizar. El primero que debemos hacer notar es la evidencia que justifica el problema de la investigación, a saber: Necesidad de obtener una buena clasificación de imágenes de plantas de albahaca cultivadas con diferentes concentraciones de nitrógeno, utilizando una base de datos de 144 imágenes de entrenamiento y 72 imágenes de validación por cada clase evaluada utilizando modelos de Redes Neuronales de gran capacidad del estado del arte, como son VGG16 y ResNet50V2, con inicialización de pesos en la base de datos ImageNet. Como era de esperar los resultados obtenidos, evaluados en la métrica de la exactitud que alcanza el modelo, son malos. La principal razón la podemos encontrar en el escaso número de imágenes de la base de datos de llegada, el aprendizaje por transferencia y la técnica clásica de aumento de datos mediante las transformaciones que se realizan en el Image Generator, no son suficiente para ofrecerle a la redes la capacidad de aprender la distribución de características propias de las clases que componen la base de datos. El segundo resultado de interés es que la aparente mejor clasificación del modelo VGG sobre el modelo ResNet5[V2], que contradice el resultado que podemos esperar en correspondencia con lo discutido de ambos modelos, se sustenta en un aprovechamiento de características del fondo de la imagen, no del área de la planta, que hace este modelo. Esto queda en evidencia cuando los resultados de ambos modelos se igualan en valor promedio para la exactitud de todos los split con la imagen de la planta sobre un fondo negro. Por último vale destacar que la diferencia del valor de la métrica evaluada para el split 2, en comparación con los otros dos en el entrenamiento de ambos modelos, para las imágenes sobre el fondo original y entre todos los split, para los dos modelos cuando se trabaja sobre fondo negro, hacen pensar que la distribución de las imágenes en cada split fija características propias a cada uno de ellos

condición esta que no debería existir. Todos los split deberían tener la capacidad de generalizar todas las características de todos los posibles datos de entrada. Aquí nuevamente podemos fijarlo como una consecuencia del número pequeño de imágenes de la base de datos de llegada.

6.5. Gráficas del Entrenamiento Clásico.

Las gráficas correspondientes al proceso final de ajuste fino en el entrenamiento de transferencia clásico evidencian variaciones importantes de los valores obtenidos para la exactitud y la pérdida (accuracy y loss) entre todas las épocas que se suceden hasta el final del entrenamiento, esto evidencia que la estabilidad de los parámetros de los modelos no se alcanza y estos son muy sensibles a las variaciones de los datos de entrada.

En la Figura(6.3) se muestran las gráficas del entrenamiento para el modelo ResNet50V2 en los ejemplos de mayor valor del accuracy alcanzado correspondientes al Split1.

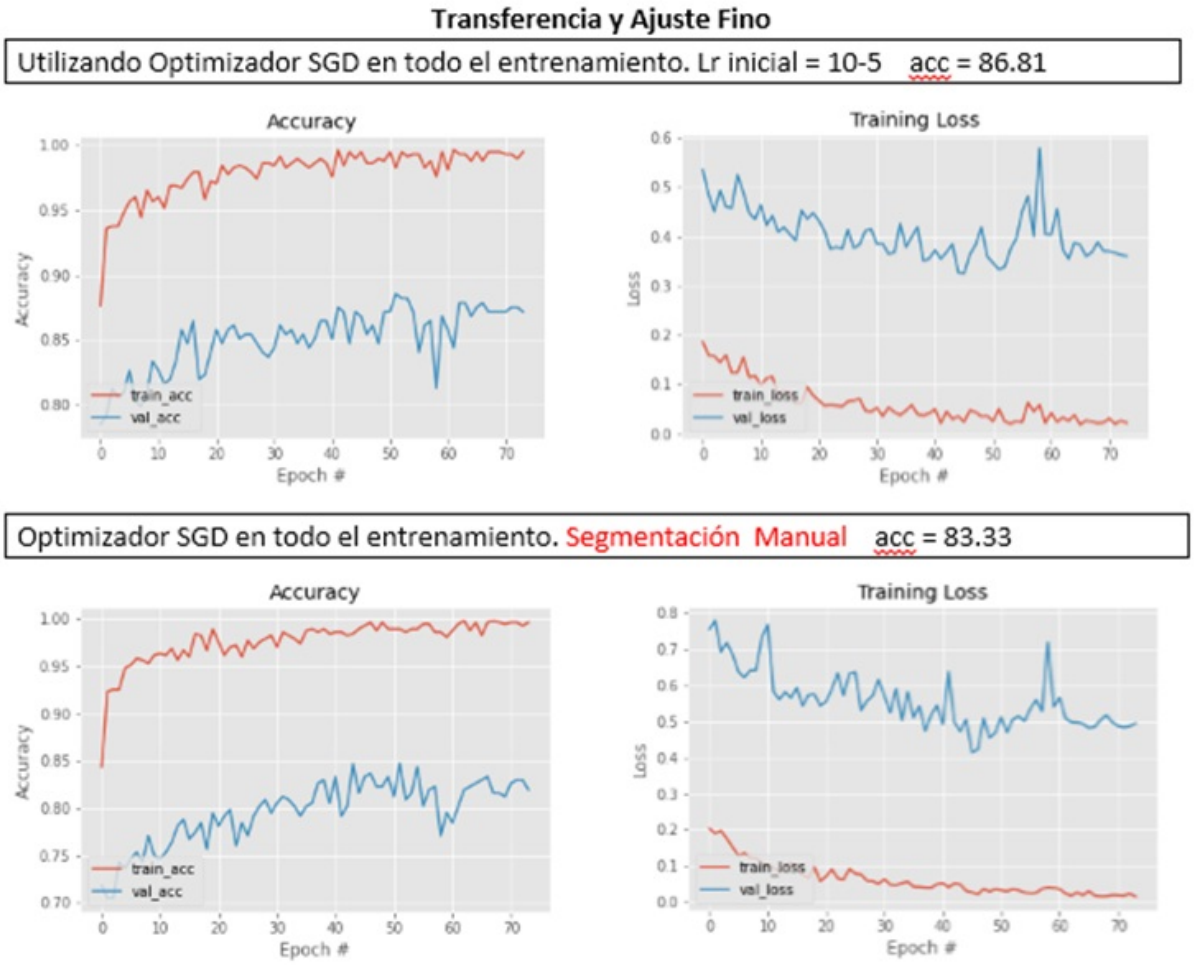


Figura 6.3: Gráficas del entrenamiento clásico para el modelo ResNet50V2.

6.6. Entrenamiento Curricular.

La Tabla (6.2) muestra los valores obtenidos en el entrenamientos curricular de los dos modelos de estudio, VGG16 y ResNet50V2. Al igual que la tabla con los resultados del entrenamiento clásico muestra el valor alcanzado para la exactitud de los modelos cuando se entrenan con las imágenes sobre el fondo original y con segmentación manual de la planta sobre un fondo negro.

Tabla 6.2: Resultados del entrenamiento curricular en los dos modelos del estudio.

Transferencia de Aprendizaje y Ajuste Fino Optimizador SGD		
Imágenes de la planta sobre el fondo original Accuracy (%):		
	VGG16	ResNet50V2
Split1	92.593 ± 0.454	86.574 ± 0.818
Split2	77.315 ± 1.772	82.176 ± 1.772
Split3	91.319 ± 3.119	93.287 ± 0.818
Promedio	87.076 ± 1.782	87.346 ± 1.136
Imágenes con segmentación manual de la planta sobre un fondo negro		
Split1	77.546 ± 4.468	84.144 ± 3.630
Split2	76.620 ± 4.346	86.806 ± 1.801
Split3	85.880 ± 2.788	88.426 ± 1.588
Promedio	80.015 ± 3.867	86.459 ± 2.340

En este caso el análisis de los resultados nos permite apreciar la mejora que ofrece el método de entrenamiento implementado para ambos modelos y por tanto la veracidad de la hipótesis formulada: Fijar el entrenamiento del modelo de red bajo el enfoque curriculum-by-smoothing garantiza mejores valores de los parámetros que caracterizan el aprendizaje. El valor de la exactitud promedio que alcanzan los modelos mejora hasta un valor superior al 87 %, cuando se entrenan en las imágenes sobre el fondo original. El resultado del entrenamiento es mejor en todos los split, para los dos casos de estudio. Es de destacar que para el modelo de red ResNet50[V2] la mejora en el valor promedio de esta métrica es de 7.4 %. Al tener este modelo un mayor número de capas convolucionales es de esperar que el efecto producido por la utilización de los filtros a la salida de las mismas sea mayor, potenciado además por la presencia de los bloques residuales en la estructura del modelo. Esta afirmación también se sustenta con los resultados obtenidos en el entrenamiento curricular utilizando las imágenes con fondo negro, en estas condiciones VGG16 solo alcanza mejorar 0.3 % en le valor promedio de exactitud del modelo sin embargo, ResNet50[V2] se mantiene con una mejora cercana al valor anterior alcanzado (en este caso 7.1 %).

6.7. Evaluación de la precisión del modelo entrenado a través de la matriz de confusión.

En la investigación que desarrolla este trabajo la observación del comportamiento de la matriz de confusión durante el entrenamiento curricular implementado permite apreciar la progresión de los aumentos de los aciertos de la red a la hora de clasificar en las diferentes clases a las imágenes de validación por los modelos entrenados en los diferentes valores de sigma en que se fijan los filtros y de esta forma validar la influencia favorable que establecen la sucesión de inicializaciones que estos filtros experimentan para el valor final alcanzado en la exactitud del modelo entrenado. (Figura (6.4))

6.8. Gráficas del Entrenamiento Curricular.

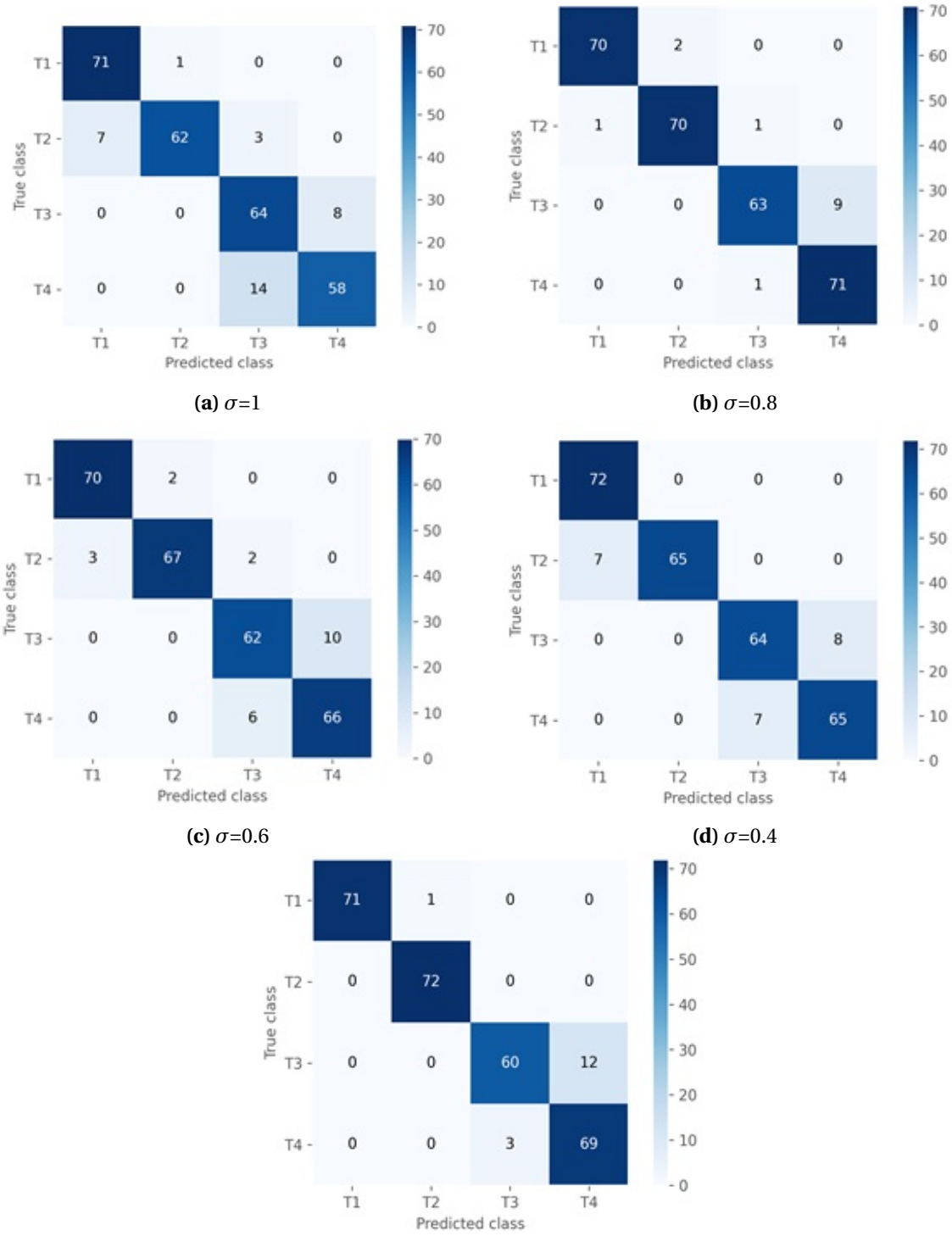
Las gráficas finales del entrenamiento curricular evidencian un comportamiento menos variable de los valores medidos que los mantiene casi iguales durante todas las épocas, después de un ajuste de pocas épocas al inicio del entrenamiento.

En la Figura(6.5) se muestran las gráficas del modelo ResNet50[V2] en el ejemplo de mayor valor de exactitud alcanzado correspondientes al Split1, en la última etapa del entrenamiento donde el valor de σ para los filtros es 0.2 .

También se muestran en Figura (6.6) las gráficas de un entrenamiento curricular completo, donde cada una representa el entrenamiento hasta un máximo de 50 épocas para un valor específico del parámetro σ fijado en las capas de filtro gaussiano. En este caso se muestran las gráficas del modelo ResNet50V2 en el Split3.

6.9. Resultados para las imágenes con segmentación automática de la planta.

El trabajo para obtener las imágenes con segmentación manual de las plantas en las imágenes con precisión a nivel de píxel es engorroso y obliga un consumo de tiempo grande por el evaluador. Una manera más sencilla de evaluar la influencia del fondo, en la clasificación de las imágenes en las que se entrena el modelo, es la realización de la segmentación del área de interés de manera automática. Una forma de hacerlo es, como se menciona anteriormente, transformar las imágenes RGB al espacio de color HSV (tono, saturación y valor) usando la biblioteca OpenCV [OpenCV] y detectar aquellas regiones de píxeles que pertenecen a la planta especificando valores máximos y mínimos para cada canal. A continuación se muestran los resultados obtenidos en el entrenamiento de los modelos aplicando previamente esta transformación a la base de datos de entrenamiento y validación Tabla (6.3) y la comparación de los resultados promedio del entrenamiento curricular por suavizado de los tres splits en los que se dividió la base de datos con las imágenes con fondo natural y sobre un fondo negro Tabla(6.4).



(a) σ=1 **(b)** σ=0.8

(c) σ=0.6 **(d)** σ=0.4

(e) σ=0.2

Figura 6.4: Resultado de las matrices de confusión para el modelo VGG16 con entrenamiento curricular. Val-Acc-Final: 0.9444.

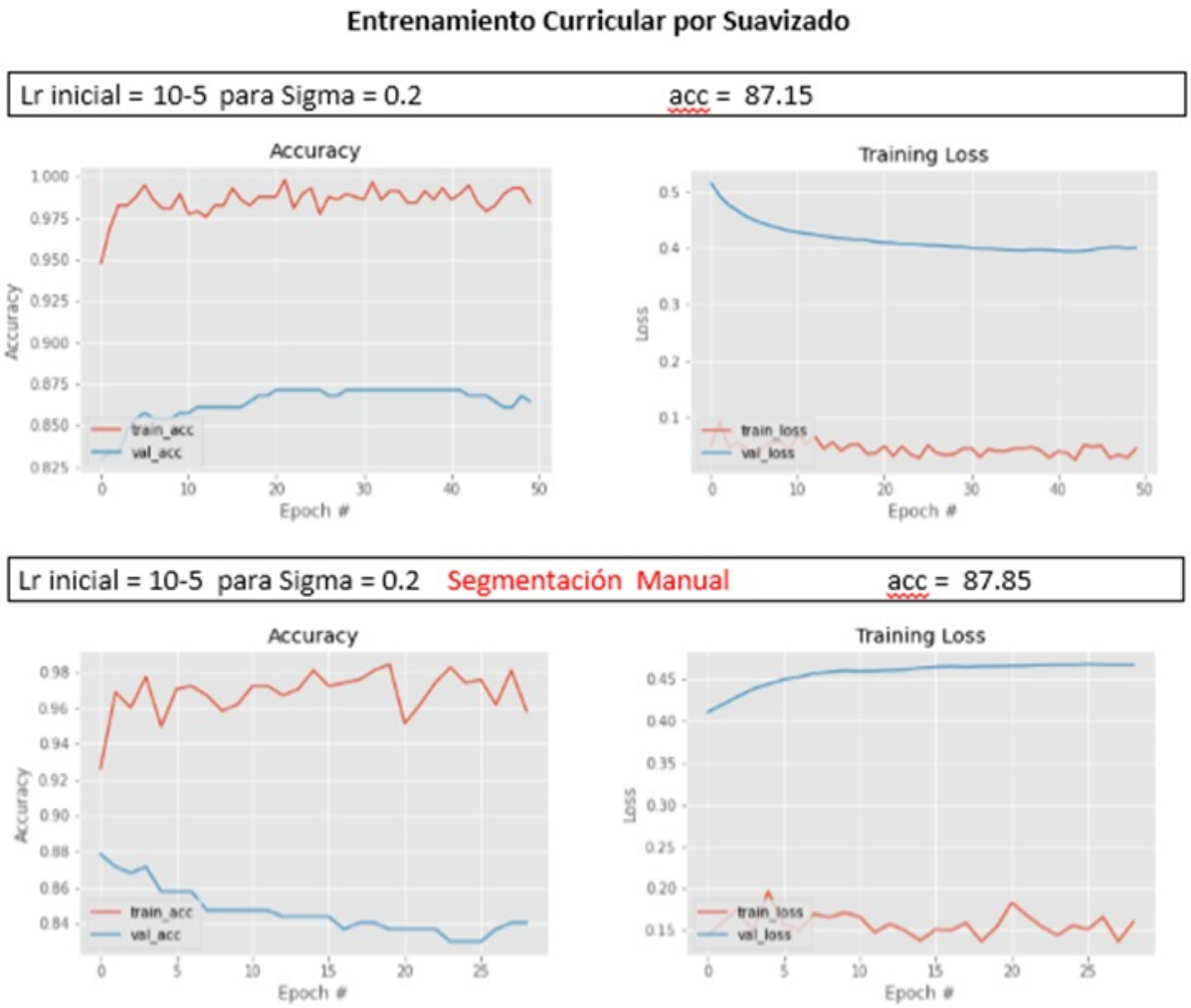
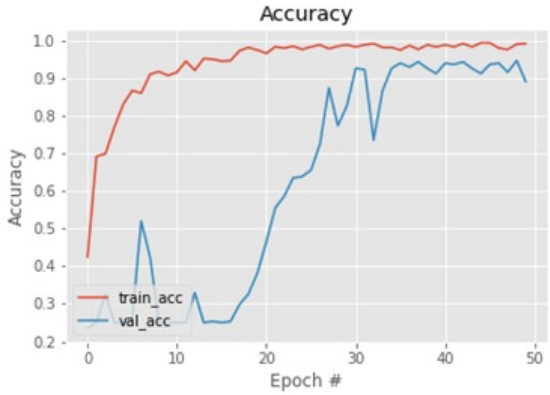


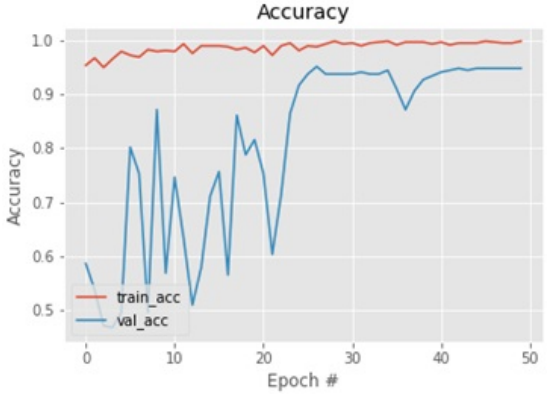
Figura 6.5: Gráficas del entrenamiento curricular para el modelo ResNet50V2.

Tabla 6.3: Resultados para las imágenes con segmentación automática de la planta.

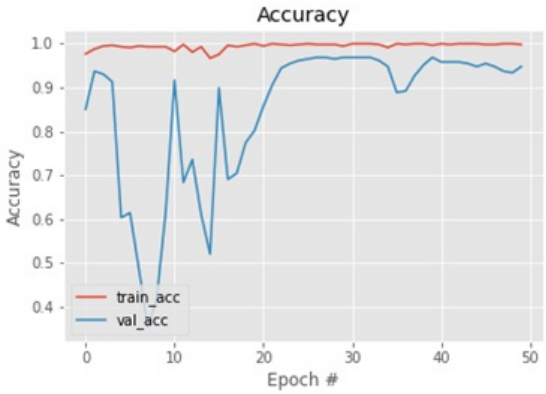
	Optimizador SGD	
	VGG16	ResNet50V2
	Accuracy (%)	
Imágenes con segmentación automática de la planta sobre un fondo negro		
Split1	84.722 ± 5.138	88.889 ± 3.492
Split2	79.745 ± 3.341	86.343 ± 2.975
Split3	86.806 ± 3.216	90.162 ± 1.263
Promedio	83.757 ± 3.898	88.465 ± 2.577



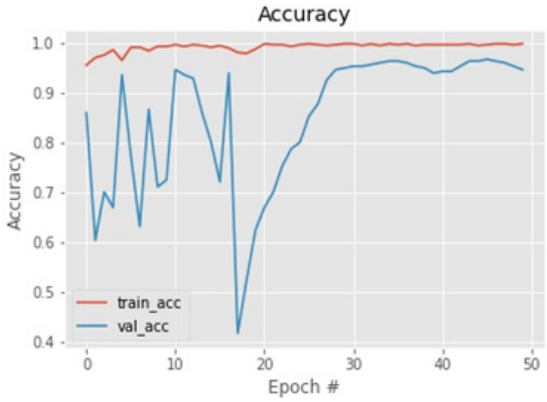
(a) σ=1



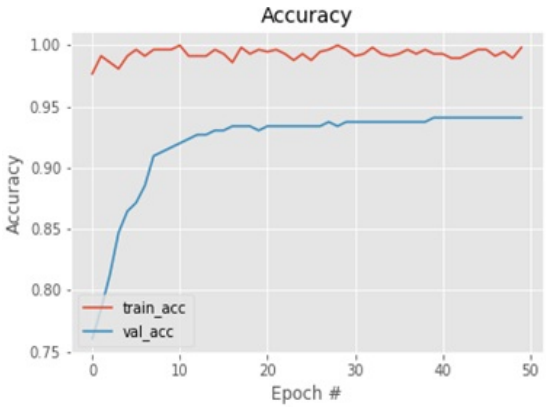
(b) σ=0.8



(c) σ=0.6



(d) σ=0.4



(e) σ=0.2

Figura 6.6: Gráficas de un entrenamiento curricular para el modelo entrenado en los diferentes valores de σ.

Tabla 6.4: Comparación de los resultados promedio del entrenamiento curricular.

	Optimizador SGD	
	VGG16	ResNet50V2
	Accuracy (%)	
Promedio (Imágenes Originales)	87.076 ± 1.782	87.346 ± 1.136
Promedio (Segmentación Automática)	83.757 ± 3.898	88.465 ± 2.577
Promedio (Segmentación Manual)	80.015 ± 3.867	86.459 ± 2.340

Capítulo 7

Conclusiones

✓ El análisis cuantitativo de los resultados demuestra la obtención de un mayor valor de exactitud en la clasificación de las imágenes de validación para ambos modelos cuando seguimos un método de entrenamiento curicular por suavizado, en comparación con los valores que se alcanzan en la validación siguiendo un entrenamiento de transferencia clásico.

✓ Las características apreciadas en las gráficas de los entrenamientos curriculares muestran un rápido acceso hacia un valor más óptimo de los parámetros y su estabilidad alrededor de ese valor durante las restantes épocas del entrenamiento en la última inicialización de los filtros, tanto para el set de entrenamiento como para el de validación, lo cual nos indica la obtención de una mejor capacidad de generalizar para los modelos entrenados.

✓ De manera general en las imágenes analizadas, el entrenamiento curricular termina con áreas GradCam, señaladas en las imágenes de validación, más pequeñas y mejor ubicadas sobre la zona de interés (la planta). Dicha reducción del área de activación visible, parece relacionarse con un aumento del valor de clasificación de la imagen como perteneciente a una clase específica.

✓ Se aprecia una progresión en aumento de los aciertos de la red a la hora de clasificar las imágenes de validación en correspondencia con la sucesión de valores de sigma en los que son inicializados los filtros gaussianos, lo que valida el esquema curricular propuesto para todo el entrenamiento.

Bibliografía

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S. & Asari, V. K. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, *8*(3). https://www.mdpi.com/2079-9292/8/3/292

Anderson, J. A., Silverstein, J. W., Ritz, S. A. & Jones, R. S. (1977). *Distinctive features, categorical perception, and probability learning: Some applications of a neural model*. Psychological Review.

Barrios, A. (2004). Filtrado [Universidad de Valladolid]. https://slideplayer.es/slide/1737320/

Bengio, Y., Louradour, J., Collobert, R. & Weston, J. (2009). Curriculum Learning, 41-48. https://doi.org/10.1145/1553374.1553380

Brownlee, J. (2020). *Use Early Stopping to Halt the Training of Neural Networks At the Right Time*. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1800-1807. https://doi.org/10.1109/CVPR.2017.195

D. Rumelhart, G. H. & Williams, R. (1986). *Learning representations by back-propagating errors*. Nature.

Data-Science-Team. (2021). *Función de activación Relu*. https://datascience.eu/es/aprendizaje-automatico/funcion-de-activacion-relu/

EcuRed. (2009). Procesamiento digital de imágenes. https://www.ecured.cu/Procesamiento_digital_de_im%C3%A1genes

Elman, J. L. (1993). Learning and development in neural networks: the importance of starting small. *Cognition*, *48*(1), 71-99. https://doi.org/https://doi.org/10.1016/0010-0277(93)90058-4

Filtros digitales [Apuntes de la asignatura Teledetección en Geografía. Universidad de Murcia]. (2003). https://www.um.es/geograf/sigmur/teledet/tema06.pdf

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A. & Brendel, W. (2018). *Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness*. https://openreview.net/forum?id=Bygh9j09KX

Gonzalez, R. C. & Wintz, P. (1996). *Procesamiento digital de imágenes. Addison-Wesley* [Tema 3,4, pág 89-269].

Gonzalez, R. C. & Woods, R. E. (2008). *Digital image processing* [http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X]. Prentice Hall.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning* (M. Press, Ed.). http://www.deeplearningbook.org

Haykin, S. (2009). *Neural Networks and Learning Machines* [https://books.google.com.mx/books?id=KCwWOAAACAAJ]. Pearson. http://dai.fmph.uniba.sk/courses/NN/haykin.neural-networks.3ed.2009.pdf

He, K., Zhang, X., Ren, S. & Sun, J. (2015a). *Deep Residual Learning for Image Recognition*. arXiv: 1512.03385 [cs.CV]. https://arxiv.org/abs/1512.03385

He, K., Zhang, X., Ren, S. & Sun, J. (2015b). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Hebb, D. O. (1949). *The Organization of Behavior: A neuropsychological theory*.

Hernández Hernández, A. (2020). *Visualización e Interpretación de Redes Neuronales Convolucionales mediante Dropout Espacial*. https://riull.ull.es/xmlui/bitstream/handle/915/21778/Visulazacion%20e%20interpretacion%20de%20redes%20neuronales%20convolucionales%20mediante%20dropout%20espacial..pdf?sequence=1

Hinton, G. E. (2009). *Deep belief networks*. http://scholarpedia.org/article/Deep_belief_networks

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*, 107-116. https://doi.org/10.1142/S0218488598000094

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hopfield, J. & Tank, D. (1985). *Neural computation of decisions in optimization problems*. Biological Cybernetics.

ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). (2014). https://image-net.org/challenges/LSVRC/2014/

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K. & Geras, K. (2020). The Break-Even Point on Optimization Trajectories of Deep Neural Networks. https://arxiv.org/abs/2002.09572

Keras API reference / Callbacks API. (2021). https://keras.io/api/callbacks/

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097-1105.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989). *Backpropagation Applied to Handwritten Zip Code Recognition*. https://doi.org/10.1162/neco.1989.1.4.541

M. Ancona, C. O., E. Ceolini & Gross, M. (2017). Unified View of Gradient-Based Attribution Methods for Deep Neural Networks. In Proceedings of the NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning [California, USA].

M. Ribeiro, S. S. & Guestrin, C. (2016). Why Should I Trust You?: Explaining the Predictions of Any Classier. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics [California, USA].

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models [Computer Science Department, Stanford University, CA 94305 USA]. https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf

Martín López, R. (2019). *Detección de personas en imágenes de profundidad mediante redes neuronales convolucionales*. https://ebuah.uah.es/dspace/handle/10017/38433?locale-attribute=es

McCulloch, W. & W., P. (1943). *A logical calculus of the ideas immanent in nervous activity*.

Minsky, M. & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.

Muneeb ul, H. (2018). *VGG16 – Convolutional Network for Classification and Detection*. https://neurohive.io/en/popular-networks/vgg16/

Neyshabur, B., Sedghi, H. & Zhang, C. (2021). What is being transferred in transfer learning?

Ortega, B. R., Biswal, R. R. & DelaCruz, E. S. (2019). Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial. https://rcs.cic.ipn.mx/2019_148_7/Deteccion%20de%20enfermedades%20en%20el%20sector%20agricola%20utilizando%20Inteligencia%20Artificial.pdf

Pastor Martín, D. P. (2018). *Usando Redes Neuronales Convolucionales Para Convertir Características Visuales en Estímulos Sonoros*. https://http://riull.ull.es/xmlui/handle/915/10422

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier.

Rivera, M. (2019). *La Red Residual (Residual Network, ResNet)*. http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/resnet/resnet.html

Rodriguez, I. (2020). *Efecto de la sustitución de la función de Pooling en Redes Neuronales Convolucionales*. https://academica-e.unavarra.es/bitstream/handle/2454/38674/Iosu%20Rodr%C3%ADguez%20Mart%C3%ADnez%20-%20TFM.pdf?sequence=1&isAllowed=y

Rojas, R. (2018). *Multimedia, Unidad de Competencia IV. Imagen*. http://ri.uaemex.mx/bitstream/handle/20.500.11799/103085/secme-35716_1.pdf?sequence=1&isAllowed=y

Ron Kohavi, G. H. J. (1997). Wrappers for feature subset selection. https://www.researchgate.net/profile/Ron_Kohavi/publication/243768287_Wrappers_for_fea%20ture_selection/links/55a6b4d108ae51639c573eef/Wrappers-for-feature-selection.pdf

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y

Schmidhuber, J. & Hochreiter, S. (1997). *Logocode*. Inst. für Informatik. https://books.google.com.mx/books?id=ols-GwAACAAJ

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D. (2019). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, *128*(2), 336-359. https://doi.org/10.1007/s11263-019-01228-7

Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

Sinha, S., Garg, A. & Larochelle, H. (2021). Curriculum By Smoothing. https://arxiv.org/abs/2003.01367

Smith, S. W. & Ph.D. (2002). *The Scientist and Engineer's Guide to Digital Signal Processing*. (2.ª ed.). http://www.dspguide.com/ch6.htm

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15*(56), 1929-1958. http://jmlr.org/papers/v15/srivastava14a.html

Stanford, C. (2021). CS231n: Convolutional Neural Networks for Visual Recognition [Stanford-Course]. https://cs231n.github.io/transfer-learning/

Thrun, S. (1996). *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. https://www.amazon.com/Explanation-Based-Neural-Network-Learning-International/dp/0792397169

Torres, J. (2019). *DEEP LEARNING Introducción práctica con Keras*. https://bookskcjwwk.blogspot.com/2021/01/descargar-deep-learning-introduccion.html

Torrey, L. & Shavlik, J. (2010). *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* [https://www.igi-global.com/book/handbook-research-machine-learning-applications/486]. IGI global.

Wang, C.-F. (2018). *A Basic Introduction to Separable Convolutions*. https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728