Clasificación de Flores con Redes Neuronales Convolucionales

Yalila Aljure Jiménez

Especialista en Analítica y Ciencia de Datos

Tutor
Javier Fernando Botia Valderrama
Ph.D. Ingeniería Electrónica

Universidad de Antioquia
Facultad, Departamento de Ingeniería de Sistemas
Posgrado en Analítica y Ciencia de Datos
Medellín, Colombia
2021.

Cita	(Aljure Jiménez, 2021)
Referencia **Estilo APA 7 (2020)**	Aljure Jiménez, L. (2021). *Clasificación de Flores con Redes Neuronales Convolucionales* [Trabajo de grado especialización]. Universidad de Antioquia, Seleccione ciudad UdeA (A-Z).

Especialización en Analítica y Ciencia de Datos, Cohorte II.

Línea de Investigación:

Redes Neuronales Convolucionales

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes

Decano/Director: Jesús Francisco Vargas Bonilla

Jefe departamento: Diego José Luis Botia Valderrama.

Resumen

En los últimos años las redes neuronales artificiales (ANN) han tenido un crecimiento muy importante dentro del área de aprendizaje automático gracias a los avances tecnológicos. Generación de grandes bases de datos, mejores modelos y nuevas ideas para el mejoramiento continuo de los diferentes algoritmos.

Una de las formas más utilizadas de las ANN son las redes neuronales convolucionales (CNN), las cuales son utilizadas generalmente para la clasificación de imágenes. Este documento busca clasificar una base de datos que contiene imágenes de cinco clases de flores, utilizando tres arquitecturas de CNN muy conocidas: VGGNet, Xception y DenseNet.

Los resultaron mostraron que la arquitectura Xception generaba los mejores resultados de clasificación alcanzando una exactitud de 0.803, una precisión de 0.8201 y una sensibilidad de 0.7798.

Palabras claves: Clasificación, Aprendizaje Automatizado, Redes Neuronales Convolucionales.

Contents

1. Introducción

Las redes neuronales artificiales (ANN) son procesos computacionales inspirados en el sistema nervioso humano. Son un numeroso conjunto de nodos que se interconectan para aprender colectivamente de los datos de entrada y así poder generar un óptimo resultado. Normalmente, los datos de entrada son vectores multidimensionales, los cuales serán distribuidos a lo largo de las capas ocultas de la red.

Las redes neuronales convolucionales (CNN) son análogas a las redes neuronales artificiales, donde cada neurona recibe un insumo y realiza una operación, la red neuronal seguirá generando unos pesos que al final se traducirán en la efectividad del resultado. En el caso de clasificación de imágenes el resultado se verá reflejado en la eficiencia de la red para clasificar imágenes nuevas.

La única notable diferencia entre las ANN y las CNN, es que las CNN tienen unas arquitecturas que están especialmente diseñadas para trabajar la codificación de imágenes. Estas arquitecturas se componen básicamente de tres tipos de capas; las capas convolucionales, las capas de muestreo ("*pooling*") y las multicapas ("*fully connected*"). Es importante mencionar que la creación y optimización de este tipo de modelos puede tomar un buen tiempo y sus resultados pueden ser confusos. (Nash, 2015)

En los últimos años las CNNs han demostrado ser muy eficientes a la hora de clasificar imágenes de gran escala y videos. Algunas de las arquitecturas que más influenciaron este crecimiento son LeNet, AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet y DesNet. (Musab COŞKUN1, 2017)

El objetivo de este trabajo es clasificar un conjunto de imágenes de flores de cinco clases diferentes utilizando las CNNs. A pesar de que el conjunto de datos se encuentra balanceado a lo largo de las cinco clases y que las imágenes pertenezcan a una categórica, la base de datos solo posee 3,669 imágenes, de tamaño diferente y con algo de ruido presente.

Para el desarrollo de este trabajo, primero se utilizó la arquitectura VGG16. La cual fue una mejora de la arquitectura AlexNet al incorporar más capas y demostrar que esto mejoraba el rendimiento del modelo. (Zisserman K. S., 2015)

Luego de varias iteraciones y lograr unos buenos resultados con la arquitectura VGG16, se utiliza la arquitectura Xception y la Desnet-121. La Xception es una adaptación de la arquitectura Inception. Este tipo de arquitecturas trabajan en párelo y buscan reducir el problema de desvanecimiento del gradiente descendente(Chollet F. , 2017). La DenseNet es una red más profunda que la Xception y la VGG-16, que ha demostrado ser más eficaz y más eficiente a la hora de hacer los entrenamientos.

El documento esta organizado de la siguiente forma: primero se hace un resumen conceptual sobre el aprendizaje automatizado, las redes neuronales y en particular las redes neuronales convolucionales. Luego, se hace una reseña histórica de la clasificación de imágenes

utilizando redes neuronales. Adicionalmente, se revisa que se ha desarrollado hasta el momento en temas relacionados con clasificación de imágenes de flores.

En el tercer capítulo de este documento, se introduce la metodología que se va utilizar. En el cuarto capítulo, se presentan los diferentes resultados y sus respectivos análisis. Finalmente, se presentan las diferentes conclusiones y las posibles mejoras que se pueden hacer para mejorar los resultados aquí encontrados.

2. Marco Teórico

2.1. Marco Conceptual

Desde el inicio de los computadores, los especialistas han buscado replicar el comportamiento de los humanos. Son muchas las formas, en que una maquina puede simular comportamientos inteligentes, sin embargo, se pueden clasificar en dos categorías débil y fuerte. Débil, son aquellos sistemas que pueden cumplir con un conjunto de tareas limitado, mientras el comportamiento fuerte, por el contrario, son sistemas que se pueden aplicar a una gran variedad de tareas. (Schank, 1991)

El termino de Inteligencia Artificial (I.A.) fue utilizado por primera vez en 1956, por John McCarthy, Marvin Minsky y Claude Shannon en la Conferencia de Dartmouth, un congreso en el que se hicieron previsiones triunfalistas a diez años que jamás se cumplieron, lo que provocó el abandono casi total de las investigaciones en estos temas durante quince años.

Andreas Kaplan y Michael Haenlein definen la inteligencia artificial como «la capacidad de un sistema para interpretar correctamente datos externos, para aprender de dichos datos y emplear esos conocimientos para lograr tareas y metas concretas a través de la adaptación flexible». (Andreas Kaplan, 2018)

Dentro de los campos de la I.A., está el aprendizaje automatizado ("*Machine Learning")*, el cual busca dotar a las máquinas de capacidad de aprendizaje. Existen varios tipos de algoritmos de aprendizaje, donde los más conocidos son: supervisado, no supervisado y por refuerzo.

<u>Aprendizaje Supervisado</u>

El aprendizaje supervisado tiene por objetivo crear una función capaz de predecir el valor numérico o etiqueta, este último si es un modelo de clasificación, de un objeto de entrada.

Para resolver un problema de aprendizaje supervisado (Figura 1), el primer paso es determinar qué tipo de datos se van a utilizar, para luego hacer una preparación y un preproceso de los mismos.

La selección del algoritmo es el paso más crítico, el cual se realiza después de haber definido el conjunto de datos de entrenamiento. Existen varios tipos de algoritmos como los basados en lógica, los basados en la noción de perceptrón o redes neuronales, los de aprendizaje estadístico, los basados en instancias, las máquinas de soporte vectorial de soporte (SVM) por sus siglas en inglés, entre otros.

Una vez determinar la función que va a resolver el problema, se ejecuta el algoritmo de aprendizaje, se evalúa y se clasifica.

Los algoritmos de aprendizaje supervisado tienen una muy amplia gama de aplicaciones. En aplicaciones financieras, de imagen y vídeo, para clasificar y rastrear objetos; en aplicaciones industriales, para detectar valores atípicos; en mantenimiento predictivo, para estimar la vida

útil de equipos industriales; en aplicaciones biológicas, para detectar tumores y descubrir fármacos; en aplicaciones de energía eléctrica, entre muchas otras.

Aprendizaje No Supervisado

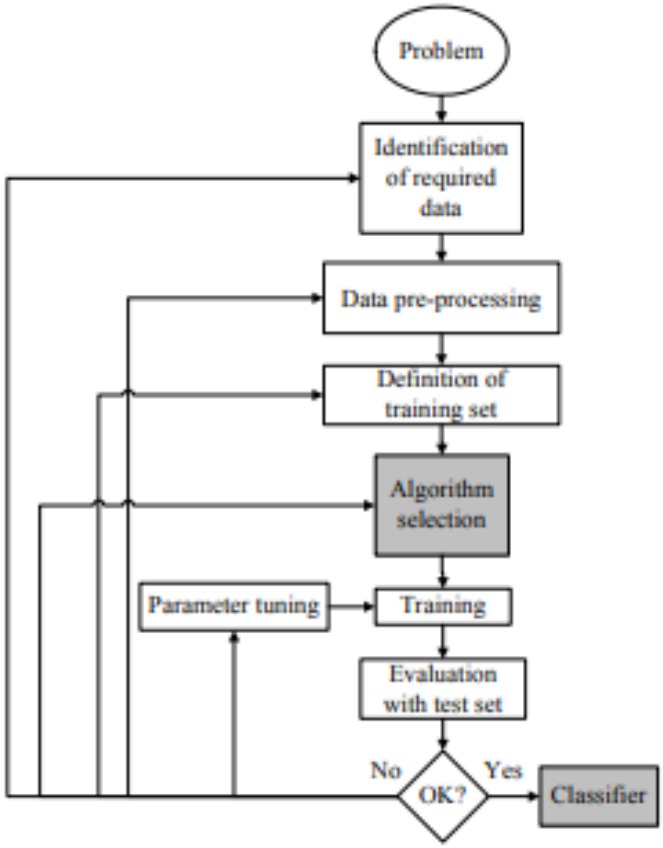
Este es un tipo de aprendizaje donde el modelo se ajusta a las observaciones, no hay un conocimiento a priori como en el caso del aprendizaje supervisado. Se descubre en los datos de entrada las características, regularidades, correlaciones y categorías. El entrenamiento de este tipo de algoritmos suele ser menor frente al supervisado.

Algunas de las aplicaciones del aprendizaje no supervisado son:

- Segmentación de conjuntos de datos por atributos compartidos.
- Detección de anomalías que no encajan en ningún grupo
- Simplificación de conjuntos de datos agregando variables con atributos similares

Figura 1.

Proceso Supervisado



Nota. Adaptado de ¨*Supervised Machine Learning: A Review of Classification Techniques, por Informatica*¨ (Kotsiantis, 2007).

Aprendizaje Reforzado

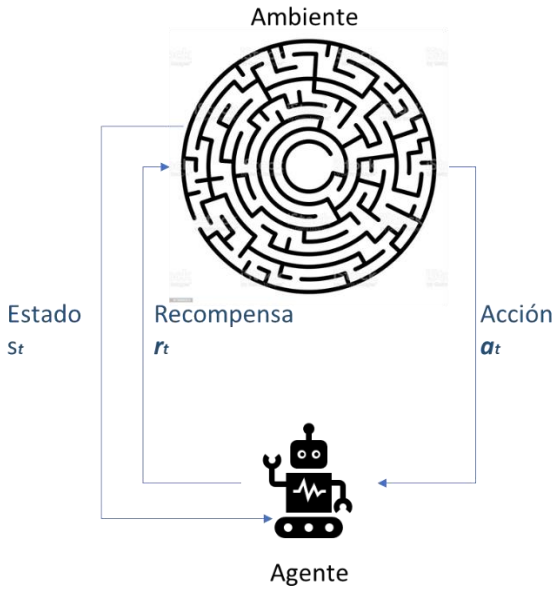
Este es el tipo de aprendizaje es el más general entre las tres categorías. Es un sistema que básicamente aprende a base de ensayo y error. En este caso no hay un agente que indique que se debe hacer, el agente inteligente debe aprender a comportarse mediante un entorno de recompensas y castigos. El objetivo principal es maximizar la señal de recompensa para tomar las mejores decisiones posibles.

En un aprendizaje por refuerzo estándar, un agente está conectado a un ambiente por medio de percepción y acción (Figura 2). En cada interacción el agente recibe como entrada una indicación de su estado actual ($s \in S$) y selecciona una acción ($a \in A$). La acción cambia el estado y el agente recibe una señal de recompensa ($r \in R$).

Las principales aplicaciones de los algoritmos por refuerzo se desarrollan dentro de la resolución de problemas.

Figura 2.

Proceso Reforzado



Nota. La figura es una representación de un proceso reforzado.

Aprendizaje Profundo

El aprendizaje profundo ("*Deep Learning*") es un subcampo del aprendizaje automatizado, una nueva interpretación de los datos que pone énfasis en el aprendizaje sucesivo a través de capas. La palabra profundo se relaciona con el número de capas, por lo general este tipo de modelos cuenta con miles de capas.

Esta representación de las capas se deriva de los modelos de redes neuronales artificiales, el cual hace referencia a la neurobiología. De hecho, muchos de los conceptos del aprendizaje profundo vienen del entendimiento del cerebro humano. Son capaces de aprender automáticamente diferentes tipos de representaciones y abstracciones como imágenes, video, sonido y texto.

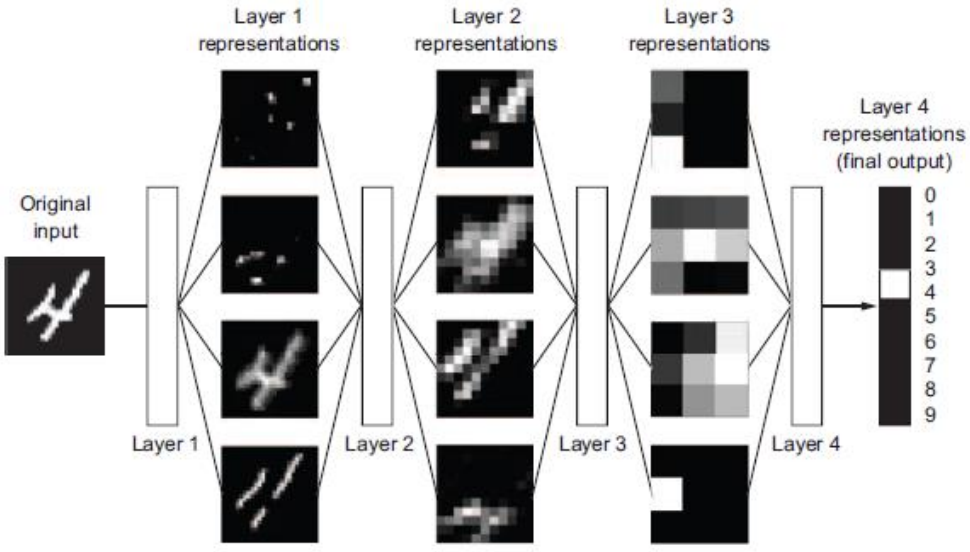
La arquitectura básica de una red neuronal es una capa de entrada, varias capas escondidas y una capa de salida, y consta de dos procesos uno de propagación hacia adelante ("*Forward Propagation*") y otro de propagación hacia atrás ("*Back Propagation*").

En la propagación hacia delante comienza con una capa de entrada que se conecta a las siguientes capas escondidas a través de varios canales que tienen asignados unos pesos. Inicialmente, estos pesos se asignan aleatoriamente, generando una primera clasificación que seguramente está muy alejada de la realidad. Pero con cada ejercicio de entrenamiento los pesos se van ajustando y la perdida va disminuyendo. El ajuste lo realiza el optimizador en la propagación hacia atrás y es lo que lo convierte en el elemento clave del aprendizaje. Ese ciclo de entrenamiento se repetirá hasta minimizar la función de costo y se consiga el resultado deseado.

En la figura 3, se puede observar la representación gráfica de una red neuronal que clasifica dígitos del 0 al 9. La red transforma un digito en varias representaciones que a lo largo de las capas se va diferenciado más de la imagen original, sin embargo, se acerca más a una realidad informativa que permite una clasificación adecuada de la imagen de entrada. De esta forma, una capa es una transformación de la imagen inicial (*Input*) que es almacenada en unos pesos (*wi*).

Figura 3.

Red Neuronal Artificial para la clasificación de dígitos.



Nota. Deep Learning with Python, (Chollet F. , 2017)

En la práctica, los algoritmos de aprendizaje profundo son neuronas interconectadas que se organizan en capas, lo que las hace diferentes es el tipo de arquitectura y forma de entrenamiento que tengan. Las principales clases de redes neuronales por sus siglas en ingles son la MLP (Perceptrón Multicapa), CCN (Redes Neuronales Convolucionales) y RRN (Redes Neuronales Recurrentes).

Conceptos Matemáticos detrás del Aprendizaje Profundo

Tensores

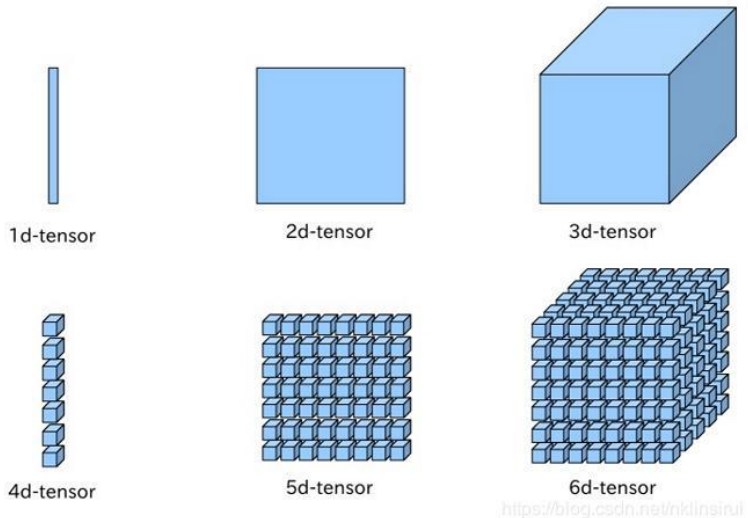
Una de las mayores ventajas de las redes neuronales frente a otro algoritmo del aprendizaje automatizado, es la capacidad de trabajar con datos no estructurados como imágenes. Para esto las redes neuronales utilizan un objeto matemático llamado tensor capaz de almacenar todo tipo de información.

Existen tensores de muchas dimensiones, en la figura 4 se presentan los tipos de tensores más utilizados en el Aprendizaje Profundo. Si, por ejemplo, tuviéramos 15 imágenes de tamaño 230 por 250 pixeles con tres canales de colores (RGB), la manera de representar esta información sería con un tensor de dimensión 4. Cabe aclarar que la literatura de aprendizaje profundo utiliza los términos rango y dimensión indistintamente, lo que a nivel algebraico estaría mal.

Los tensores son definidos por 3 atributos clásicos[1]: número de rangos, forma y tipo de datos.

Figura 4.

Ejemplos de Tensores de Varias Dimensiones.



Nota. Tomado de https://programmerclick.com/article/2724389767/

[1] (Chollet F. , 2017)

Los tensores, T, se pueden representar con:

- Transformaciones multilíneas $T(\ldots, \ldots)$ de dos formas lineales
- Sus componentes y sus bases $T = T^{ij} e_i \otimes e_j$
- Notación abstracta de índices $T \blacktriangleright T^{ij}$

Para describir las operaciones y propiedades de los tensores, utilizaremos la notación abstracta por facilidad. Las siguientes son algunas de las operaciones más comunes de los tensores:

- Suma y Resta de tensores

  Para sumar dos tensores, A y B, es importante que ambos tengan la misma base y rango:

  $$A_{ij} \pm B_{ij} = C_{ij} \qquad (1)$$

- Productos de tensores
  - Contracción

    Es básicamente contraer dos índices (uno arriba y uno abajo) y hacer la sumatoria de ellos. Un ejemplo clásico es el producto punto.

    $$A^i B_{ij} = C_j \qquad (2)$$

  - Producto Directo

    Es básicamente la multiplicación de todos los términos con todos los términos.

    $$A^i B_{jk} = C^i_{jk} \qquad (3)$$

Las siguientes son las propiedades de los tensores que están involucradas en las operaciones de aprendizaje profundo:

- Simetría y Antisimétrica

  Un tensor, T, simétrico en un par de índices si al intercambiarlos sigue dando el mismo resultado.

  $$T^{ab} = T^{ba} \qquad (4)$$

  Un tensor es asimétrico en un par de índices si al intercambiarlos sigue dando el mismo resultado, pero negativo.

  $$T^{ab} = -T^{ba} \qquad (5)$$

- Descomposición

  Un tensor puede ser descompuesto en una parte simétrica y otra antisimétrica.

$$A = Sym\ T + \text{Asym}\ T \qquad (6)$$

$$A = \begin{pmatrix} a & e & f \\ e & b & g \\ f & g & c \end{pmatrix} + \begin{pmatrix} 0 & p & -q \\ -p & 0 & r \\ q & -r & 0 \end{pmatrix} = \begin{pmatrix} a & e+p & f-q \\ e-p & b & g+r \\ f+q & g-r & c \end{pmatrix}$$

El tensor antisimétrico se (C^{ik}) se puede representar como un pseudovector (C_i):

$$C_i = \frac{1}{2}\varepsilon_{ijk}C^{ik} \qquad (7)$$

Operación Convolucional

La operación convolucional de dos tensores es básicamente la correlación cruzada de estos la cual se denota de la siguiente forma en un sistema de tiempo lineal ("*Lineal time-invariant*"), donde los valores son escalares y tiempo es discreto[2]:

$$y(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)h(\tau - 1) \qquad (8)$$

Matemáticamente lo anterior se puede expresar de la siguiente forma ya que tiene la propiedad de ser conmutativa:

$$\sum_{\tau=-\infty}^{\infty} x(\tau)h(\tau - 1) = \sum_{\tau=-\infty}^{\infty} x(t - \tau')h(\tau') \qquad (9)$$

Función de Costo

Generalmente, en los procesos de aprendizaje supervisado la función de costo se conoce como el error estimado entre el valor verdadero y las predicciones, de manera que queremos minimizar este valor a lo largo del proceso de optimización.

Las funciones de costo más utilizadas en los procesos de aprendizaje profundo son:

[2] (Benoit Liquet, n.d.)

- Error cuadrado medio (MSE), representado como la diferencia promedio del valor estimado ($\widehat{y_i}$) y el valor real (y_i) al cuadrado:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2 \qquad (10)$$

Esta medida funciona muy bien en regresiones lineales.

- Error absoluto medio (MAE), representado de la siguiente forma:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y_i} - y_i| \qquad (11)$$

- Entropía cruzada binaria $\iota(\theta)$), calcula que tan lejos está el valor del resultado estimado ($\widehat{y_i}$) de su valor real (y_i), comparando la probabilidad estimada (p_i) de los resultados:

$$\iota(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\left(p_i + (1-y)\log(1-p_i)\right) \qquad (12)$$

Esta medida funciona muy bien en modelos de clasificación binaria.

- Entropía cruzada categórica

$$\iota(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \log\left(p_{ij}\right) \qquad (13)$$

Esta medida es una generalización de la binaria y como su nombre lo indica funciona muy bien en los modelos de clasificación múltiple.

Gradiente Descendiente

El gradiente descendiente es la base de muchas técnicas de aprendizaje automatizado, donde se busca encontrar los parámetros de *W* (pesos de la red neural) que minimizan la función de costo. De manera que el gradiente descendiente es un método de optimización numérica que busca reducir el error estimando los mejores coeficientes encontrados.

El gradiente descendiente se puede calcular como las derivadas parciales respecto a los parámetros θ:

$$\frac{\delta J(\theta)}{\delta W} \qquad (14)$$

Donde $J(\theta)$ es la función de costo y *W* los pesos que se quieren ir actualizando en cada iteración:

$$W \leftarrow W - \alpha\frac{\delta J(\theta)}{\delta W} \qquad (15)$$

La tasa de aprendizaje α controla el tamaño de la actualización, y asegura que los cambios en W no sean demasiados grandes o demasiados pequeños, ambos extremos generarían problemas a la hora de encontrar los valores óptimos que minimizan la función de costo.

El gradiente descendiente funciona mucho mejor si los datos están escalados, ya que la forma geométrica que se recorre en el proceso de optimización tiene una escala similar en todos sus vértices haciendo que el gradiente descendiente no se demore tanto en converger.

El gradiente desciende ha tenido varias mejores a lo largo de los años, los siguientes son algunos de los mecanismos de optimización más conocidos:

- Descenso Estocástico del Gradiente (SGD)

  Esta es una aproximación estocástica más eficiente del gradiente descendiente usado para minimizar la función de costo, ya que el número de cálculos en cada iteración es menor.

  El término estocástico viene por el hecho de que, si en cada iteración, en vez de tomar la suma sobre toda la población, solo se hace sobre una muestra. Para luego calcular el gradiente como el promedio de los gradientes de la muestra.

  Algunas de las desventajas de este gradiente están relacionadas con que requiere una serie de hiperparámetros y que es muy sensible a la escala de las características.

- Adam

  El optimizador Adam trata de solventar el problema con la fijación de la ratio de aprendizaje del SGD, para ello adapta la ratio de aprendizaje en función de cómo estén distribuidos los parámetros. Si los parámetros están muy dispersos ("*sparse*") la tasa de aprendizaje aumentará.

- Adagrad

  El optimizador Adagrad es un algoritmo basado en el gradiente descendiente, que adapta la ratio de aprendizaje a los parámetros. Lo que lo hace muy conveniente cuando el modelo tiene varias dimensiones.

  El Adagrad realiza grandes actualizaciones cuando los parámetros son poco frecuentes y pequeñas actualizaciones cuando son muy frecuentes.

Además de los optimizadores mencionados aquí, existen otros que terminan siendo también variantes mejoradas del gradiente descendiente como Adedelta, Ademax, Nadam, RMSProp, entre otros. Para seleccionar el optimizador hay que considerar sus particularidades y como estas se ajustan al problema que se quiere resolver.

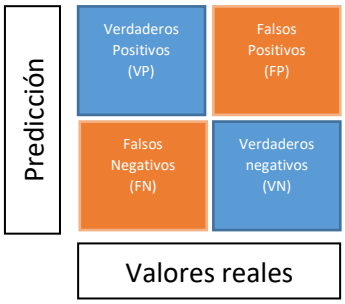
Métricas de Error

- La matriz de confusión

  Es una herramienta que permite analizar el desempeño de un algoritmo. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa la instancia real de esa clase.

Figura 4.

Matriz de Confusión Binaria.



Nota. Representación gráfica de la matriz de confusión.

- Exactitud ("*Accuracy*")

  Esta métrica se refiere a lo cerca que esta el resultado de una medición con su verdadero valor. Es la suma de las predicciones correctas (verdadero negativo (VN) y verdadero positivo (VP)) dividido por el número total de predicciones del modelo (verdadero negativo (VN), verdadero positivo (VP), Falso Negativo (FN) y Falso Positivo (FP)):

$$\frac{VN+VP}{VN+VP+FN+FP} \qquad (16)$$

Ahora bien, la Exactitud puede no ser la medida de error más correcta ya que esta no distingue entre los dos tipos de errores (Falsos Negativos (FN) y Falsos Positivos (FP)).

- Precisión ("*Precision*")

  Esta métrica se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuando menor es la dispersión mayor es la precisión y se calcula como:

$$\frac{VP}{VP+FP} \qquad (17)$$

- Sensibilidad (*"Recall"*)

  Se conoce como la tasa de verdaderos positivos y nos está diciendo que tan bueno es el modelo a la hora de discriminar los casos positivos. Se calcula como:

$$\frac{VP}{VP+FN} \qquad (18)$$

- Especificidad

  Se conoce como la tasa de verdaderos negativos y nos está diciendo que tan bueno es el modelo a la hora de discriminar los casos negativos y se calcula como:

$$\frac{VN}{VN+FP} \qquad (19)$$

- F1 score

  Esta es otra métrica muy empleada porque nos resume la precisión y la sensibilidad en un solo cálculo: 2 * (Sensibilidad * Precisión) / (Sensibilidad + Precisión).

Al final, la conveniencia de usar una métrica u otra dependerá de cada caso en particular.

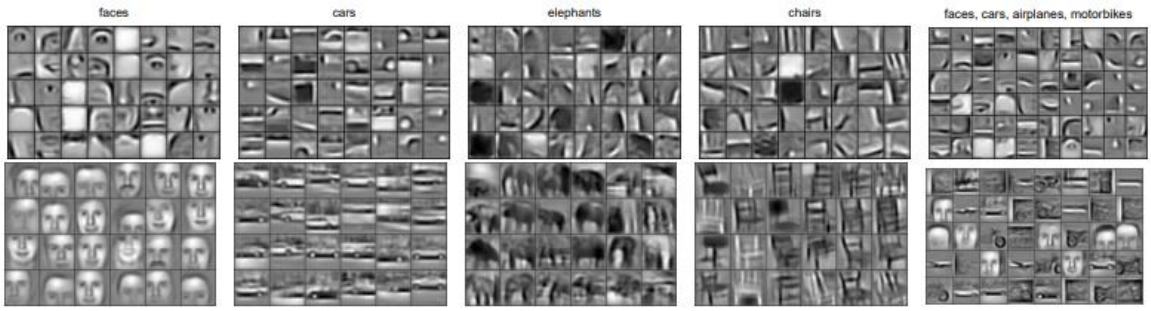
Redes Neuronales Convolucionales

Un tipo de aprendizaje profundo muy utilizado para la clasificación de imágenes son las redes neuronales convolucionales o CNN por sus siglas en inglés (*"Convolutional Neural Network"*).

La idea detrás de las CNN es que una imagen puede descomponerse en elementos claves. Cada capa va aprendiendo diferentes niveles de abstracción. Una primera capa aprende elementos básicos como aristas, una segunda capa ya aprende patrones compuestos de elementos básicos y así sucesivamente cada capa aprende patrones más complejos (Figura 5).

Figura 5.

Niveles de abstracción de una red convolucional.



Nota. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. (Honglak Lee, 2009).

La arquitectura de una CNN está formada por los siguientes tipos de capas: convolucional, agregación ("*Pooling*"), densa ("*Fully Connected*"), y puede incluirse capas de normalización y abandono ("*Dropout*") dependiendo de las necesidades de modelo.

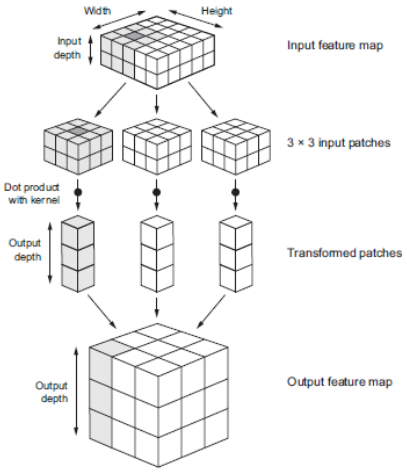
Capa Convolucional

La idea intuitiva detrás de una capa convolucional es detectar características o rasgos visuales en las imágenes. Esta es una propiedad muy interesante ya que una vez la red aprende cierta característica es capaz de reconocer esta misma característica en cualquier otro punto de la imagen. Esto permite que las redes convolucionales aprendan eficientemente conceptos visuales cada vez más complejos y abstractos.

En general las redes convoluciones operan sobre un tensor 3D que almacena las características de la imagen (Altura, Ancho, Color) y un tensor *W* que representa los filtros, uno para cada característica que se quiera detectar.

Figura 6.

Capa convolucional.



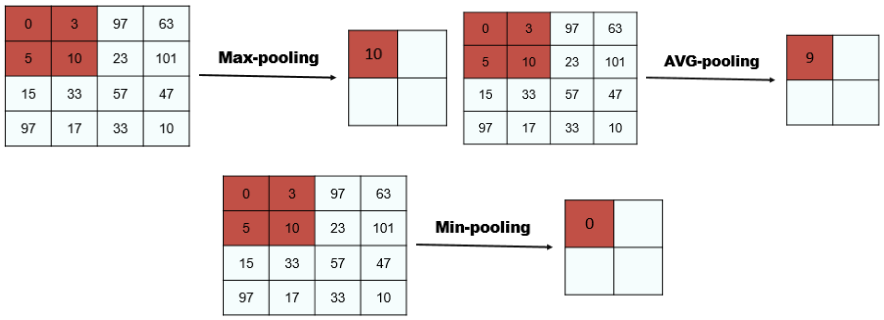
Nota. *Deep Learning with Python* (Chollet F. , 2017)

Capa de Agrupación

En las arquitecturas de las CNN la capa convolucional viene acompañada de una capa de agrupación, que básicamente busca condensar la información contenida en la capa anterior. Hay varias formas de condesar esta información las más conocidas son: agrupación por máximo (“*max-pooling”)*, por mínimo (“*min-pooling”)*, y agrupación por promedio (“*average-pooling”)*.

Figura 7.

Tipos agrupación.



Nota. Tomado de https://www.reachiteasily.com/2021/02/convolutional-neural-network.html

Como se observa en la figura 7, a pesar de la transformación de agrupación mantiene la proporción de la imagen.

Capa de Activación

En redes neuronales la función de activación de un nodo define la salida del mismo dado un conjunto de entradas. Las funciones de activación deben cumplir con las siguientes propiedades: ser no lineales, el rango es finito y derivable.

Existe una gran variedad de funciones de activación, a continuación, se numeran algunas de las más conocidas (Figura 8).

1. ReLU

   Esta transformación activa un nodo so la entrada está por encima de cierto umbral.

2. Sigmoid

   Esta transformación permite reducir los valores extremos o atípicos. Su rango de salida esta entre 0 y 1.

3. Softmax

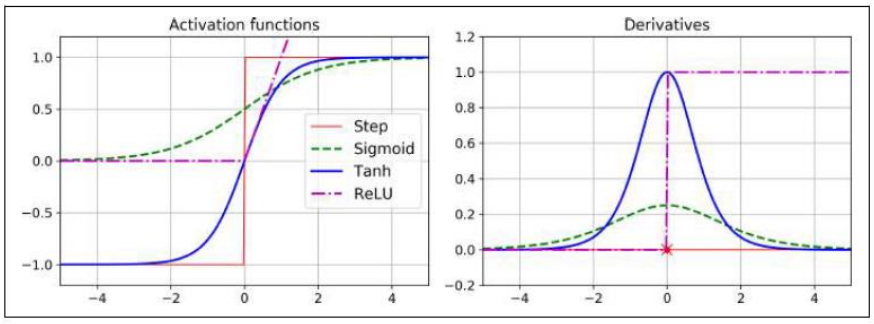
   Esta es una generalización de la regresión logística, ya que en lugar de clasificar en binario puede contener múltiples límites de decisión. Esta función de activación se encuentra a menudo en la capa de salida de la red neuronal.

4. Tanh

   A diferencia de la función de activación sigmoide, esta función de activación tiene su rango de salida entre -1 y 1.

Figura 8.

Funciones de activación y sus derivadas.



Nota. Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow. (Géron, 2019)

A pesar de que la capa de activación no suma parámetros a la red neuronal, requiere del uso del algunos hiperparametros.

Hiperparametros de la arquitectura CNN

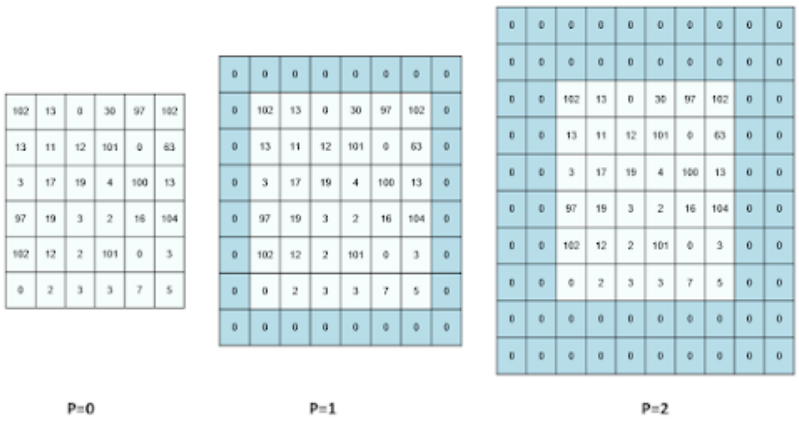
De los anteriores conceptos se desprende que los principales hiperparámetros en las arquitecturas de las CNN son el tamaño y numero de filtros, el paso de avance ("*stride*") y el relleno ("*padding*").

El tamaño del filtro permite controlar el entorno de la imagen y el número de filtros indica que características se quieren manejar.

Una vez realizada la convolución de una imagen, puede que el resultado sea un tensor de un tamaño menor, si se desea un resultado con las mismas dimensiones se utiliza el hiperparámetro de relleno, el cual consiste en agregar ceros alrededor de la imagen de entrada antes de hacer la operación de convolución con el tensor de filtros.

Figura 9.

Relleno de 1 y de 2.

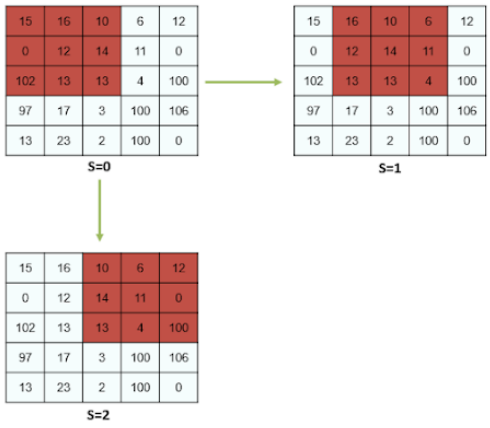


Nota: Tomado de https://www.reachiteasily.com/2021/02/convolutional-neural-network.html

Otro hiperparametro que se puede especificar en la arquitectura de las CNN es el paso de avance (*stride*) en la ventana de los filtros. Como es de esperarse un mayor avance hará que sea menor la información que pasa a la siguiente capa.

Figura 10.

Paso de avance en la operación convolucional.



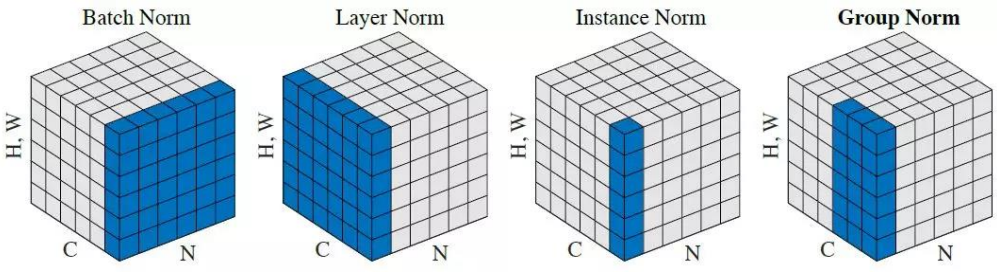
Nota: Tomado de https://www.reachiteasily.com/2021/02/convolutional-neural-network.html

Capa de Normalización y Abandono ("*Dropout*")

La capa de normalización es una técnica introducida en el 2015, la cual normalizar las entradas de una capa buscando prevenir la dispersión del gradiente. Existen varias formas de normalizar (Figura 11) y la mejor manera de normalizar depende de la arquitectura que se esté trabajando. Por lo general, las CNN normalizan por lotes ("*Batch"*), ya que por lo general la misma neurona recibirá la entrada de todas las imágenes.

Figura 11.

Paso de avance en la operación convolucional.



Nota: Tomado de https://www.it-swarm-es.com/

Por otro lado, la técnica de abono es muy utilizada para ayudar a mitigar el sobreajuste del modelo. Como su nombre lo indica consiste en abandonar cierto número de neuronas de manera aleatoria durante la fase de entrenamiento, es decir, que estas neuronas no se tienen en cuenta durante la iteración ayudando al modelo aprender y no a memorizar la generación de los resultados.

2.2. Estado del Arte

El aprendizaje profundo es un campo del aprendizaje automático que se viene estudiando desde 1950 y solo se vio prominente a principios del 2010. En pocos años ha logrado alcanzar resultados muy favorables en áreas como la clasificación de imágenes, el reconocimiento del lenguaje, transcripción de escritura a mano, asistente de digitación, entre muchos otros. (Neha Sharma, 2018)

El primero en ganar premios por hacer clasificación de imágenes haciendo uso de las redes neuronales profundas con unidades de procesamiento GPU fue Dan Cirsan en 2011. Pero el momento decisivo llegó en 2012, con la clasificación de imágenes a gran escala (*ImageNet)*, cuando Alex Krizhevsky y Geoffrey Hinton (Alex Krizhevsky, 2012) fueron capaces de alcanzar una precisión del 83.6%. Ya para el 2015 esta precisión fue de 96.4% y el problema de clasificación de las imágenes de ImageNet fue considerado un problema resuelto.

Las razones de este despegue en los últimos años por parte del aprendizaje profundo se deben en gran parte a que son modelos más eficientes y a que la ingeniería de datos se hace de manera automatizada. Adicionalmente, tres grandes fuerzas técnicas han dirigido los avances en el aprendizaje automático, el crecimiento de la información (gran volumen de datos), los avances en algoritmos y la mejora de los dispositivos computacionales.

El aprendizaje profundo tiene varias propiedades que lo ayudaran a permanecer en el tiempo como son la simplicidad en el entrenamiento de los datos, la escalabilidad del aprendizaje, lo versátiles y reusables de este tipo de modelos.

Evolución de la arquitectura de las redes neuronales convolucionales

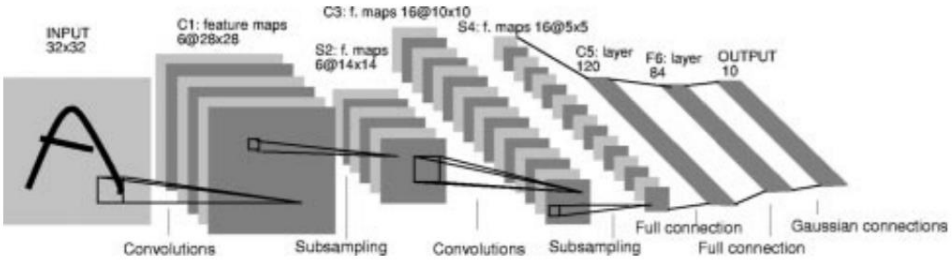
Si solo se utilizara una capa conectada a cada píxel de una imagen, no sería una arquitectura muy eficiente ya que los pixeles están correlaciones espacialmente. De manera, que se hace necesario extraer características particulares de las imágenes, y es aquí donde aparecen las redes neuronales convolucionales.

LeNet

La primera arquitectura para clasificar documentos escritos a mano. En la arquitectura las características de la imagen van siendo reducidas en la medida que se va avanzando en la red (Figura 12). Imaginemos la imagen de una casa, los primeros filtros de la red se enfocarán en la forma general de la casa, en la medida que se va avanzando, los filtros se empiezan a enfocar en características más específicas como las ventanas o la puerta de la casa.

Figura 12.

Arquitectura LeNet5.



Nota. Gradient-Based Learning Applied to Document Recognition (Y. LeCun, 1998).

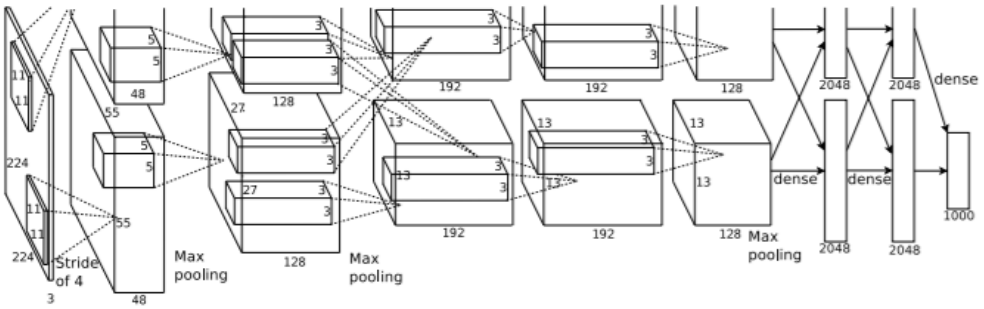
AlexNet

Así como la LeNet, esta arquitectura busca extraer características más específicas de la imagen en la medida que se avanza en la red. Sin embargo, es una red con un mayor número de capas donde se busca ya no solo clasificar imágenes 2D, sino imágenes de alta resolución con mucha más información (Figura 13).

Se introdujeron conceptos como que la función de activación no puede ser lineal, ya que la derivada es cero y esto genera problemas con el gradiente descendiente. Adicionalmente, se introduce el tema de abandono ("*Dropout*") como medida de regularización.

Figura 13.

Arquitectura AlexNet.



Nota. ImageNet Calssification with Deep Convolutional Neural networks (Alex Krizhevsky, 2012).

VGGNet

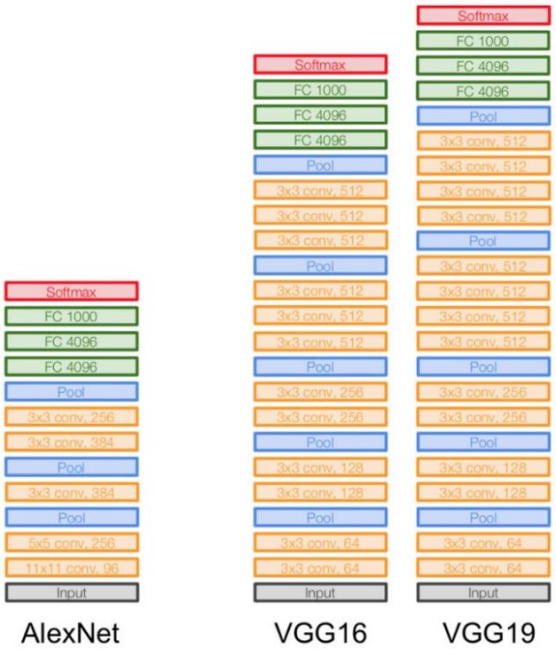
Esta arquitectura viene en la misma línea que las dos anteriores, donde se busca introducir más capas para mejorar la efectividad en la clasificación de imágenes. Esta arquitectura incluye 3 x 3 capas convolucionales completamente conectadas (Zisserman K. S., 2015).

Las motivaciones que existieron detrás del diseño de esta arquitectura fueron:

- Pequeños filtros generan mayor flexibilidad y un número menor de parámetros.
- Aglomerar las capas permite mostrar más características que no se veían inicialmente.

Figura 14.

Arquitectura Alex Net, VGG16 y VGG19



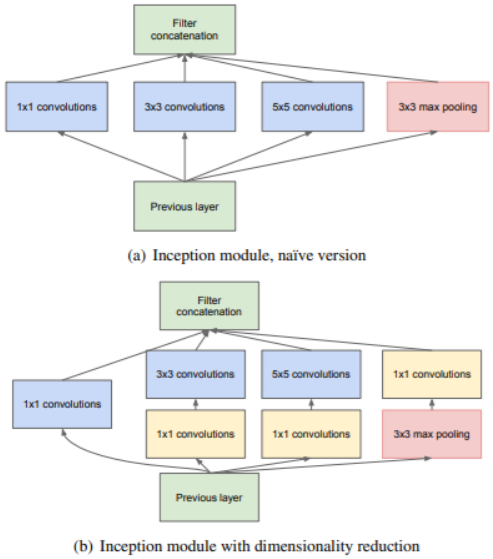
Nota: https://medium.com/deep-learning-g/cnn-architectures-vggnet

GoogleNet

El siguiente hito en clasificación de imágenes son los módulos de incepción ("*Inception"*). La idea detrás de este concepto es procesar la misma entrada en diferentes módulos convolucionales con diferentes tamaños de filtros y al final concatenarlos (Figura 15). Las mayores ventajas de este tipo de arquitecturas es que reduce la dimensionalidad del modelo y es capaz de combina múltiples características de las imágenes para ser clasificadas.

Figura 15.

Módulo de Incepción.



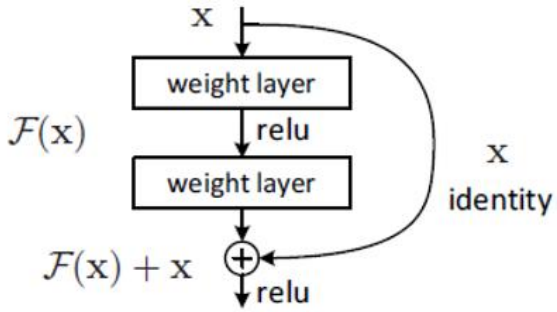
Nota. Going Deeper with Convolutions (Christian Szegedy, 2015).

ResNet

En algún punto se empieza a entender que agregar más capaz a la red no mejora su comportamiento, por el contrario, se empieza a notar un deterioro en el comportamiento del gradiente. Entonces aparece este tipo de arquitectura que incorpora un aprendizaje residual a la red (Figura 16).

Figura 16.

Aprendizaje Residual.



Nota. Deep Residual Learning for Image Recognition (Kaiming He, 2016).

En la práctica lo que busca este tipo de redes es saltarse alguna capa para moverse sobre varias capas. El salto elimina las complicaciones de la red, haciéndola más simple al usar muy pocas capas durante la etapa de entrenamiento inicial. Acelera el aprendizaje, minimizando el efecto de la desaparición de los gradientes.

Este tipo de arquitecturas resulta ser muy eficiente para la propagación, ya que el error puede propagarse por muchos caminos.

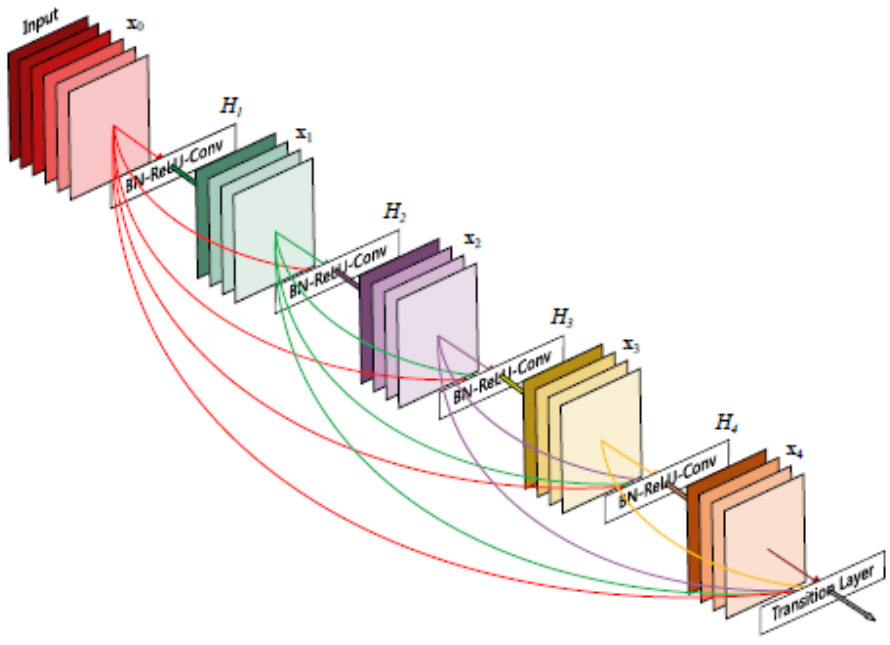
DenseNet

Finalmente aparecen las redes densamente conectadas, bloques enteros de capas conectadas unas con otras. Una red de L capas de este tipo de redes cuenta con $\frac{L*(L-1)}{2}$ conexiones, mientras que una red convencional solo cuenta con L conexiones.

A diferencia de las ResNet aquí se combinan las capas, no se suman.

Figura 17.

5 capas densamente conectadas con un k =4.



Nota. Densely Connected Convolutional Networks (Gao Huang, 2018).

Este tipo de redes tiene varias ventajas, como es: ayudar a prevenir el desvanecimiento del gradiente descendente, fortalecer la propagación de características, promover el reusó de las misma y reducir sustancialmente el número de parámetros del modelo.

Clasificación de Imágenes de Flores

Jie Zou y George Nagy, (Nagy, 2004), introdujeron el concepto CAVIAR (*"Computer Assisted Visual InterActive Recognition"*), el cual sirvió como canal de comunicación entre el humano y la máquina para reconocer elementos. Si bien la efectividad para clasificar un conjunto de imágenes es mucho mayor que dejar a la maquina sola, se toma más tiempo que el de un humano. Para este trabajo, tomaron imágenes 1078 imágenes de flores de 113 especies. Todas las imágenes tienen un tamaño estandarizado de 320 por 240 pixeles.

En el 2006, Nilasback y Zisserman (Zisserman M. N., 2006), desarrollaron un sistema de clasificación basado en un vocabulario visual, el cual representa color, forma y textura. Utilizando un modelo de clasificación SVM, alcanzaron una medida de efectividad del 72.8%.

En el desarrollo de este trabajo, Nilasback y Zisserman crearon dos conjuntos de datos que actualmente son muy utilizados para el desarrollo de modelos CNN, Oxford-17 y Oxford-102.

Oxford-17 es un conjunto de 1360 imágenes flores de 17 clases diferentes, todas originarias del Reino Unido. Son imágenes de un gran tamaño, donde la ubicación y la luz varía mucho a lo largo del conjunto de datos. Existen muchas similitudes entre diferentes clases de flores, como muchas diferencias entre las mismas clases de flores.

Oxford-102 es un conjunto de un total de 8,189 imágenes, donde cada una de las 102 categorías puede tener entre 40 y 258 imágenes. Comparado frente al conjunto de datos de Oxford-17, este conjunto de datos tiene más especies de flores, lo cual implica que existan más similitudes entre diferentes clases de flores haciendo más difícil el ejercicio de clasificación.

Ya para el 2010 Guru y colegas (D S Guru, 2010), desarrollaron un modelo KNN para clasificar 1250 imágenes en 25 especies, logrando un resultado de efectividad del 90.13%. Otras aproximaciones fueron propuestas como el de M Varma en el 2007 (Varma, 2007) o el de Z. Jie y N. George en el 2004 (Z. Jie, 2004).

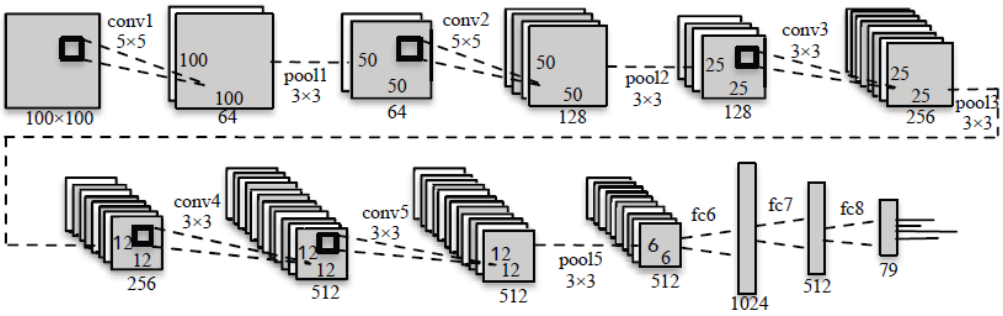
Sin embargo, en los últimos años, las técnicas de aprendizaje profundo y en especial las basadas en redes neuronales convolucionales son las que se han convertido en el estado del arte en temas de clasificación de imágenes flores, ya que han sido capaces de super cualquier resultado anterior frente a los métodos clásicos.

En el 2016, Yuanyuan Liu y un grupo de investigadores (Yuanyuan Liu, 2016), desarrollaron un modelo de redes neuronales convolucionales para clasificación de flores en dos conjuntos de datos. El primer conjunto de datos es el Oxford-102, el segundo conjunto de datos constaba de 52,775 imágenes y 79 especies de flores. La red que utilizaron tiene 8 capas, donde las primeras cinco son convolucionales y las tres restantes son capas densas. La función de activación que utilizaron en la salida es una *softmax* (Figura 18).

Los resultados obtenidos por Yuanyuan y el grupo de investigadores mostraron una eficiencia de 84.02% en el conjunto de datos Oxford-102 y 76.54% en el segundo conjunto de datos.

Figura 18.

Arquitectura de la CNN.



Nota. Flower Classification via Convolutional Neural Network (Yuanyuan Liu, 2016).

Xiaoling y Xu (Xiaoling Xia, 2017), utilizaron las características de un modelo Inception-V3 pre entrenado, para clasificar dos conjuntos de imágenes de flores (Oxford-102 y Oxford-17). Los resultaron mostraron una eficiencia del 95% para el conjunto de datos Oxford-17 y una eficiencia del 94% para el conjunto de datos Oxford-102.

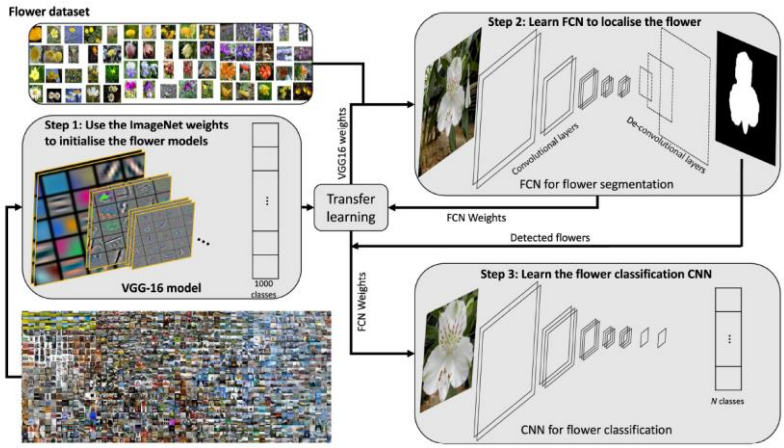
Hazem Hiary y un equipo de investigadores de la universidad de Jordania, (Hazem Hiary, 2018), desarrollaron un novedoso modelo de dos partes. Una primera parte donde segmentan la imagen de la flor utilizando un modelo de clasificación binario. En la segunda parte, se tiene una red convolucional que hace uso de las características pre entrenadas de un modelo VGG-16 (Figura 19).

Para el desarrollo de esta investigación utilizaron tres bases de datos, Oxford-17, Oxford-102 y la base de datos del trabajo de Zou-Nagy. Como es de esperarse la variabilidad que se encontró en el conjunto de imágenes de Zou-Nagy es menor que las imágenes de Oxford.

Al final, el modelo generó una efectividad de 99%, 98.5 y 97.1% para Zou-Nagy, Oxford-17 y Oxford-102, respectivamente. La efectividad de clasificación mejoro significativamente luego de usar aumentación de datos. Sin embargo, la base de datos de Oxford-102 fue el que menos mejora mostro ya que este posee el mayor número de imágenes. Por otra parte, la localización de la imagen gracias a la segmentación, las características tomadas de un modelo pre entrenado VGG-16, y la tasa de aprendizaje gradual hicieron que el resultado del modelo sea mucho más robusto que los resultados que se han presentado hasta el momento.

Figura 19.

Diagrama de Flujo del modelo de dos partes

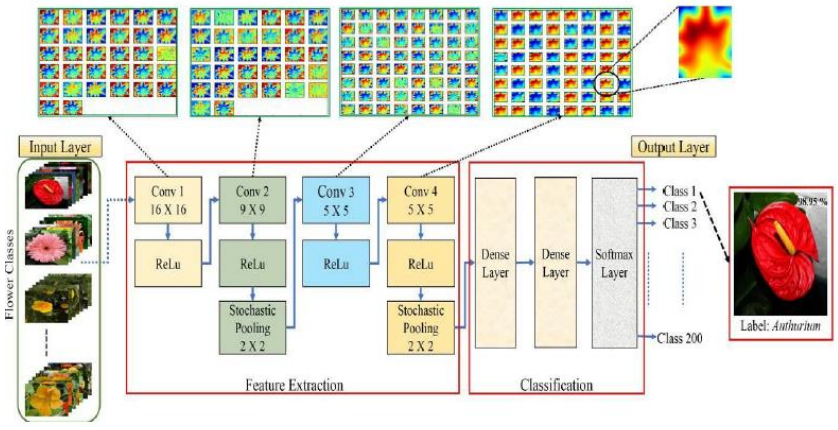


Nota. Flower Classification ussing deep convolutional neural networks (Hazem Hiary, 2018).

En el 2018, (M.V.D. Prasad, 2018), desarrollaron una arquitectura CNN para clasificar 9500 imágenes de 132 clases diferentes de flores. La red neuronal convolucional propuesta en la etapa de extracción utiliza cuatro capas convolucionales con diferentes tamaños de ventanas seguidas de una capa de activación Relu. Tres tipos de agrupamiento fueron utilizados en esta arquitectura, sin embargo, el agrupamiento estocástico fue el que genero mejores resultados. La etapa de clasificación cuenta con dos capas densas y una capa de activación (Figura 20).

Figura 20.

Red Neuronal Propuesta



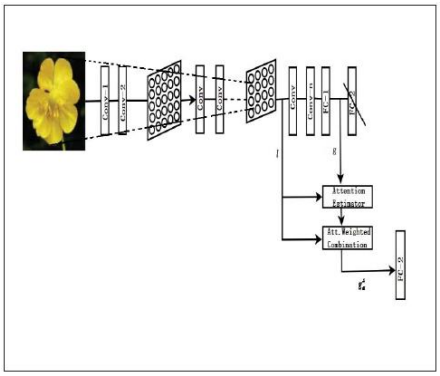
Nota. An efficient classification of flower images with convolutional neural networks (M.V.D. Prasad, 2018).

La efectividad del modelo presentado por Prassad y colegas de K L University es de 97.78%, superando la efectividad de varios estudios que se han realizado con los mismos datos.

Uno de los más recientes estudios que se tienen en el tema de clasificación de flores utilizando CNN es el Min Qin y Yuhang Xi presentado en el 2019, (Min Qin, 2019). Los autores presentan una arquitectura CNN que incluye un mecanismo de atención ("*Attention* Mechanism") y que utiliza una función de perdida de discriminante lineal (*"LD-Loss"*). El mecanismo de atención utilizada las características locales de una capa convolucional y las características globales de una capa densa (Figura 21).

Figura 21.

Módulo de atención para clasificar imágenes.

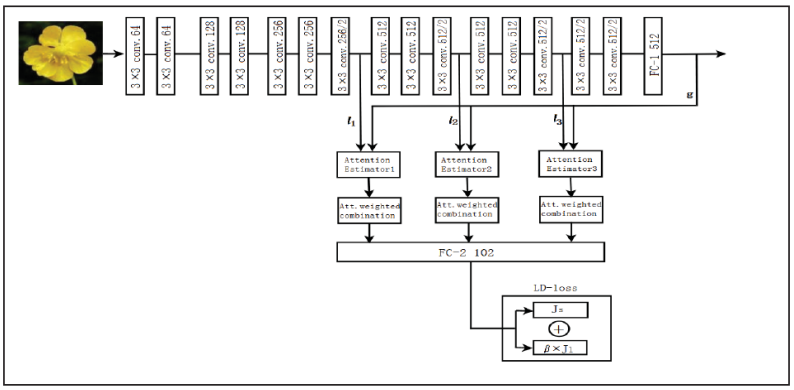


Nota. A new Improved Convolutional Neural Network Flower Image Recognition Model. (Min Qin, 2019)

El diseño de la arquitectura se divide en tres partes: una arquitectura CNN VGG-16, un mecanismo de atención y una función de perdida *LD-Loss* (Figura 22). La base de datos que utilizaron para realizar las pruebas de efectividad del diseño es la Oxford-102.

Figura 22.

Red Neuronal Propuesta.



Nota. A new Improved Convolutional Neural Network Flower Image Recognition Model. (Min Qin, 2019)

El resultado de eficacia del modelo es de 87.60%, donde las principales innovaciones fueron la introducción de un mecanismo que podía extraer características claves de las imágenes y una función de perdida donde se minimizaba las diferencias entre flores de las mismas clases y se maximiza la diferencia entre las flores de diferentes clases.

3. Metodología

A continuación, se presenta un esquema de cómo se soluciona el problema de clasificación de flores utilizando una red neuronal convencional (Figura 23).

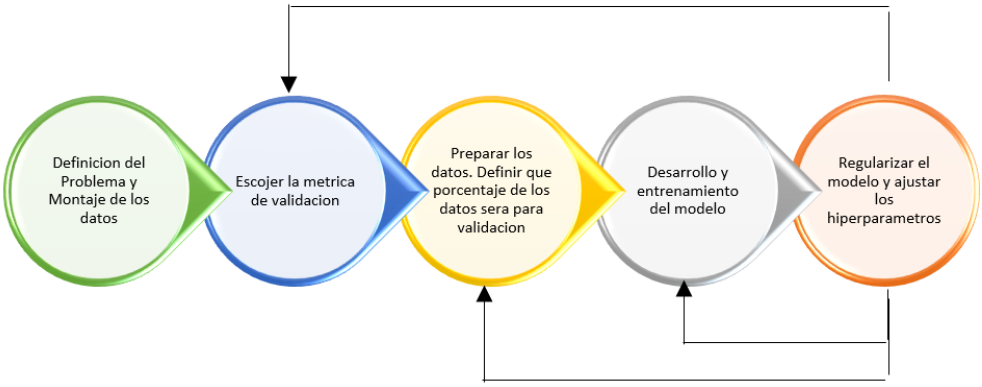
La motivación para implementar una red neuronal convolucional en la clasificación de flores, es que son modelos altamente automatizados que ayudan eficientemente a extraer características complejas de las imágenes sin laboriosos trabajos manuales.

El primer paso, fue definir el problema y entender si el conjunto de datos que se propone permite dar respuesta a la pregunta planteada. El siguiente paso, fue escoger la métrica o métricas que permitieron medir la eficacia del modelo. El tercer paso, fue preparar los datos para el entrenamiento del modelo. Finalmente, se entrenaron los modelos y se analizó la existencia de algún patrón de sobreajuste o la mejora de los resultados cambiando los hiperparametros.

Los resultados obtenidos en la última etapa permitieron entender si las métricas escogidas son las adecuadas, si el pre-procesamiento de los datos o la complejidad de la arquitectura fue la correcta.

Figura 23.

Esquema del desarrollo de la metodología.



Nota. Representación gráfica del desarrollo del trabajo

Definición del problema

Se buscó crear un modelo que clasifique de manera eficiente cinco clases de flores (margarita, diente de león, rosa, girasol y tulipán), con tamaño, forma y resolución diferentes (Figura 24).

Figura 24.

Ejemplo de la variabilidad de los datos de este estudio.

a. Similitud entre diferentes clases: i) Rose, ii) Daisy, iii) Dandelion

b. Diferencias entre de la misma clase. En este ejemplo todas son rosas.

Nota. Imágenes sacadas de la base de datos a analizar.

Las imágenes fueron descargadas del siguiente enlace de Kaggle:

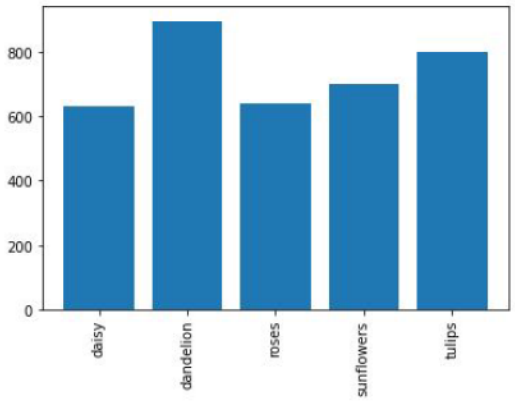
https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz

Kaggle es una de las fuentes más populares de datos públicos. Alberga importantes competencias de análisis de datos y modelado predictivo.

La base de datos para este estudio cuenta con 3,669 imágenes las cuales se encuentra balanceadas entre las 5 clases de flores (Figura 25).

Figura 25.

Balance del conjunto de datos



Nota. Datos tomados de la base de datos a analizar.

Métrica de validación

Para este estudio se consideraron las siguientes métricas de validación:

a. Exactitud ("*Accuracy"*)
b. Precisión (*"Precision"*)
c. Sensibilidad (*"Recall"*)

Estas métricas permitieron entender los resultados del modelo, no solo desde que también se clasifica las imágenes, si no desde que tan bien el modelo no está cometiendo ninguno de los dos tipos de errores (FN y FP).

Prepara los datos

1. Normalización del tamaño de las imágenes

Debido a la gran variedad que se tiene en cuanto a tamaño de las imágenes, para cada entrenamiento se normalizaron todas a un mismo tamaño. Mas adelante en este documento, se observa cómo se utilizó como tamaño estándar el tamaño mínimo de las imágenes.

2. Aumento de datos de la muestra

La cantidad de datos que se tienen para este estudio es limitada y fue necesario generar nuevos datos a partir de los datos de entrenamiento. En este estudio se aplicaron las siguientes transformaciones aleatorias para la generación de nuevos datos:

- Rango de rotación ("*rotation_range"*). Es un valor en grados entre 0° y 180°, que indica el rango dentro del cual se rotaran las nuevas imágenes al azar.

- Cambio horizontal y/o vertical ("*width_shift_range", "height_shift_range"*). Son rangos aleatorios dentro de los cuales se pueden trasladar las imágenes de manera horizontal y/o verticalmente.

- Rango de corte ("*shear_range"*). Aplica transformaciones de corte al azar sobre las imágenes.

- Rango de amplificación ("*zoom_range"*). Amplifica de manera aleatoria la imagen.

- Giro horizontal ("*horizontal_flip"*). De manera aleatoria gira horizontalmente las imágenes. El giro vertical no se tiene en cuenta en los entrenamientos de este estudio, ya que distorsionaría tanto las imágenes que perdería sentido.

- Relleno de perfiles recién creados ("*fill_mode"*). Después de hacer alguna de las transformaciones mencionadas, se requiere utilizar algún perfil de relleno que opaque la distorsión de los pixeles.

Entrenamiento del Modelo

Para la configuración y evaluación de un modelo de aprendizaje automatizado habitualmente se dividen los datos en dos conjuntos: entrenamiento ("*training"*) y prueba ("*test"*). A su vez una parte de los datos de entrenamiento se reservan para hacer validación.

Dado que el tamaño de la muestra es pequeño, se decido solo tomar datos para entrenar el modelo (80%) y datos para validarlo (20%). Adicionalmente, se fija una semilla, de manera que cada vez que se entrene el modelo se utilicen los mismos datos de entrenamiento, evitando que se filtre información de la validación y se dé un sobreajuste del modelo innecesario.

Para la clasificación de las imágenes de flores, se desarrollaron arquitecturas de redes convolucionales donde la última capa densa posee 5 neuronas, generando la clasificación de los cinco tipos de flores. Los tres tipos de arquitecturas que se plantean para el desarrollo de este trabajo son la VGG-16, la Xception y la DenseNet-121, ya que han demostrado ser arquitecturas muy eficaces a la hora de clasificar imágenes muy diversas en tamaño y forma.

Corrigiendo Sobreajustes

El sobreajuste de un modelo se produce cuando el modelo se ajusta tanto a los datos de entrenamiento que no puede realizar predicciones correctas sobre datos nunca antes vistos.

Para minimizar las situaciones de sobreajuste, se implementaron las siguientes soluciones:

- Reducir la complejidad del modelo
- Añadir capaz de abandono ("*Dropout*")
- Agregar regularizaciones L1 y/o L2
- Pare adelantado del entrenamiento ("*Earlystoping*")

Ajustes de Hiperparametros

Finalmente, se probaron diferentes hiperparámetros para encontrar la configuración óptima. Los hiperparámetros se pueden catalogar en dos grandes grupos a nivel de la red neuronal y a nivel del algoritmo.

Dentro de la red neuronal se analizó y/o modificaron los siguientes hiperparámetros:

- Numero de capas
- Numero de neuronas
- Numero de filtros y su tamaño
- Funciones de activación

En el contexto de la lógica del algoritmo de aprendizaje, se analizó y si fue preciso se modificaron los siguientes hiperparámetros:

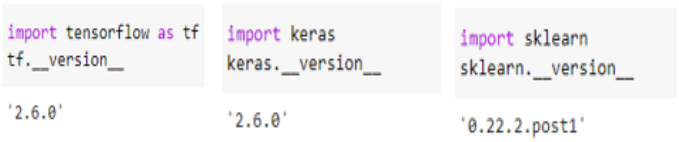
- Numero de épocas
- Tamaño del lote (“*batch*”)
- Tasa de aprendizaje
- Función de optimización y su momento

4. Resultados y Análisis

Los modelos presentados en esta sección están guardados en el siguiente link de Github: YalilaUdeA/Monografia: Clasificación de Flores con Redes Neuronales Convolucionales (github.com). El código de las implementaciones está escrito en Python. Las librerías que se utilizaron fueron Tensorflow, Keras y Sklearn (Figura 26).

Figura 26.

Versiones de las librerías



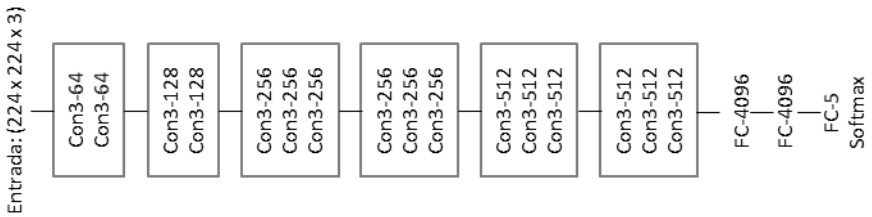
Nota. Imagen tomada de los notebooks resultados de este trabajo

Primera Iteración

Para una primera iteración se utilizó una arquitectura VGG-16 que incluyó una tasa de abandono de 0.5 en las dos últimas capas densas, una función de activación ReLU en las capaz convolucionales, y una función de activación *softmax* en la última capa de salida.

Figura 27.

Arquitectura VGG-16 utilizada



Nota. Representación gráfica de la red neuronal VGG-16 propuesta.

Se alcanzo una exactitud de 0.6731 después de entrenar el modelo a lo largo de 15 épocas.

Tabla 1

Resultados de la Primera Iteración

No. Iteración	Arquitectura	Pre - Procesamiento	Profundidad	Parámetros	Regularización	Función de Optimización	No. Épocas	Tiempo prom. por Época	Exactitud de la última época	Precisión de la última época	Sensibilidad de la última época
1	**VGG-16**	**Generación de datos:** rescale=1./255, horizontal_flip=True, vertical_flip=True, validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas Función de activación: ReLU.	33,640,005	Dropout = 0.5	SGD	15	20 seg.	**0.6731**	**No**	**No**

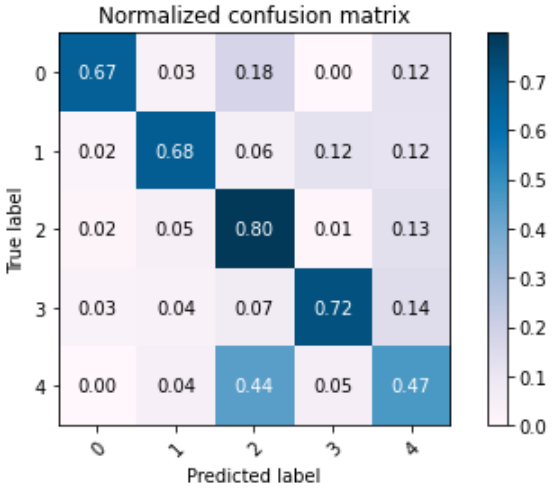
Nota. Datos resultantes de la primera iteración.

En la matriz de confusión (Figura 28) se puede observar que los casos en que más se notaron fallos del modelo, es cuando predice imágenes que son tulipanes como rosas (44%).

Por otro lado, se puede observar que en general los falsos positivos son menores que los falsos negativos (diagonal superior de la matriz). Es decir, son mayores los casos que se predice que la flor no pertenece a cierta clase cuando en realidad sí. El mayor error se observa cuando se predice que es una margarita y en realidad es una rosa (18%).

Figura 28.

Matriz de confusión de la primera iteración



Nota. Resultado de la primera Iteración. Nótese que 0 es Margarita, 1 es Diente de León, 2 es Rosa, 3 es Girasol y 4 es Tulipán.

Refinamiento del modelo convolucional

La primera modificación que se realiza es el cambio de la escala de los colores, modificando las imágenes para que la entrada del modelo sean imágenes en escala de grises. Como se observa en la tabla de resultados (Tabla 2.), en vez de obtener una mejoría, se da un deterioro en la exactitud del modelo (0.3639).

En una tercera iteración, se agregaron métodos para aumentar los datos, sin embargo, el cambio por sí solo no genero una mejora significativa frente al modelo planteado inicialmente. En las iteraciones posteriores se redujo el tamaño de las imágenes, de manera que todas las imágenes procesadas tuvieran el menor tamaño de imagen que se tiene en la base de datos. En este caso la exactitud del modelo (0.6443) no presentó mejoras frente al modelo inicial.

Se redujo la complejidad de la arquitectura (iteración 5), lo cual genero un aumento en el número de parámetros y tiempo de procesamiento del modelo, sin que esto implicara una mejora en la exactitud del mismo (0.6799).

Tabla 2.

Resultados arquitectura VGG16

No. Iteración	Arquitectura	Pre – Procesamiento	Profundidad	Parámetros	Regularización	Función de Optimización	No. Épocas	Tiempo prom. por Época	Exactitud de la última época	Precisión de la última época	Sensibilidad de la última época
2	**VGG-16**	**Generación de datos:** rescale=1./255, horizontal_flip=True, vertical_flip=True, validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: ReLU. Color: Escala de grises	33,638,853	Dropout = 0.5	SGD	15	20 seg.	**0.3639**	**No**	**No**
3	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: ReLU.	33,640,005	Dropout = 0.5	SGD	15	20 seg.	**0.6594**	**No**	**No**
4	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	Min(image_size) = 143 5 capas convolucionales + 2 densas. Función de activación: ReLU.	33,640,005	Dropout = 0.5	SGD	15	32 seg.	**0.6443**	**No**	**No**
5	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2,	Min(image_size) = 143 2 capas convolucionales + 2 densas. Función de activación: ReLU.	50,637,637	Dropout = 0.5	SGD	15	30 seg.	**0.6799**	**No**	**No**

No. Iteración	Arquitectura	Pre – Procesamiento	Profundidad	Parámetros	Regularización	Función de Optimización	No. Épocas	Tiempo prom. por Época	Exactitud de la última época	Precisión de la última época	Sensibilidad de la última época
		zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2									
6	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	Min(image_size) = 143 5 capas convolucionales + 2 densas. Función de activación: PReLU.	35,497,925	Dropout = 0.5	SGD	20	27 seg.	**0.6621**	**No**	**No**
7	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: PReLU.	38,232,133	Dropout = 0.5	SGD	60	62 seg.	**0.7332**	**0.7717**	**0.6936**
8**	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: PReLU. Batch_size = 20	38,232,133	Dropout = 0.5	SGD	100	65 seg.	**0.7880**	**0.8186**	**0.7715**
9**	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: PReLU. Batch_size = 32	38,232,133	Dropout = 0.5	SGD	100	65 seg.	**0.77**	**0.7083**	**0.6676**
10	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: PReLU. Batch_size = 32	38,232,133	Dropout = 0.5	SGD (momentum = 0.9)	100	65 seg.	**0.5622**	**0.75**	**0.3488**
11**	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: Swish. Batch_size = 32	33,740,357	Dropout = 0.5	SGD	100	60 seg.	**0.7702**	**0.7816**	**0.7538**

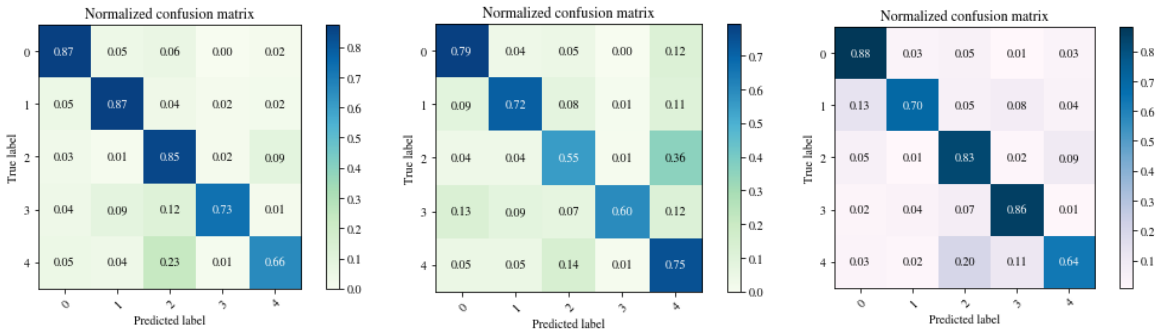
No. Iteración	Arquitectura	Pre – Procesamiento	Profundidad	Parámetros	Regularización	Función de Optimización	No. Épocas	Tiempo prom. por Época	Exactitud de la última época	Precisión de la última época	Sensibilidad de la última época
12	**VGG-16**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	image_size = 224 5 capas convolucionales + 2 densas. Función de activación: PReLU. Batch_size = 32	32,232,133	Si (L2 = 0.05)	Adam	100	64 seg.	**0.7018**	**0.9583**	**0.1888**

Nota. Datos de los resultados de las iteraciones 2 a 12.

El cambio importante ya se observa es cuando se empieza a aumentar el número de épocas y se hace un cambio en la función de activación del modelo, pasando de una función ReLU a una función PReLU, donde el resultado de la activación de los valores negativos no es cero.

Figura 29.

Matrices de confusión - VGG16



Octava Iteración **Novena Iteración** **Onceava Iteración**

Nota. Datos de los resultados de las iteraciones 8, 9 y 11.

Los mejores resultados con la arquitectura de VGG-16 se obtuvieron con una función de activación PReLU y 100 épocas (Tabla 2, iteración 8 y 9). Frente a la primera iteración, se nota una mejoría en los porcentajes de la diagonal de la matriz de confusión (Figura 28), como en los porcentajes fuera de ella.

Sin embargo, se sigue observando como las clases de rosas y tulipanes son imágenes que el modelo no ha podido diferenciar bien (Figura 30).

Figura 30.

Imágenes de Rosas y Tulipanes

Nota. Imágenes sacadas de la base de datos objeto de estudio de este trabajo.

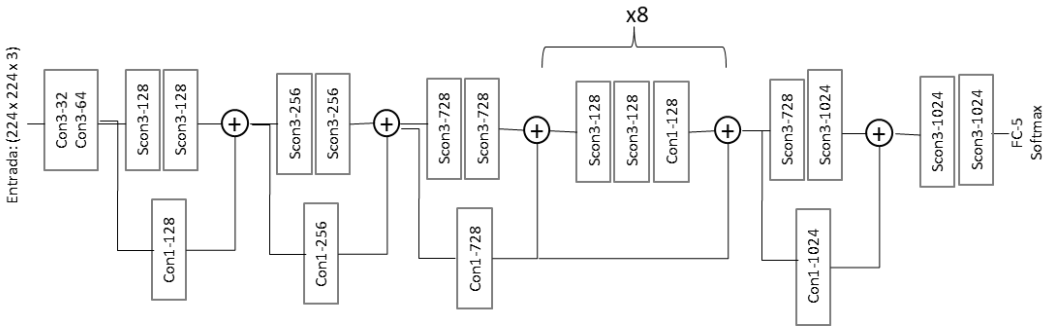
Por último, se cambió la función de activación PReLU a una función de activación *Swish* (Tabla 2, iteración 11), buscando mejorar los errores I y II. Aun así, los resultados obtenidos no superan los anteriormente mencionados.

Arquitectura Xception

Una vez se llega a un punto donde los resultados de la VGG-16 no podían mejorar, se realiza un segundo ejercicio con una arquitectura más reciente, Xception (Figura 31).

Figura 31.

Arquitectura Xception utilizada



Nota. Representación gráfica de la red neuronal Xception propuesta.

La primera gran diferencia que se observa en los resultados frente a los obtenidos con la arquitectura VGG-16, es que con la arquitectura Xception se necesitan menos épocas de entrenamiento para llegar al mismo resultado.

Tabla 3

Resultados arquitectura Xception

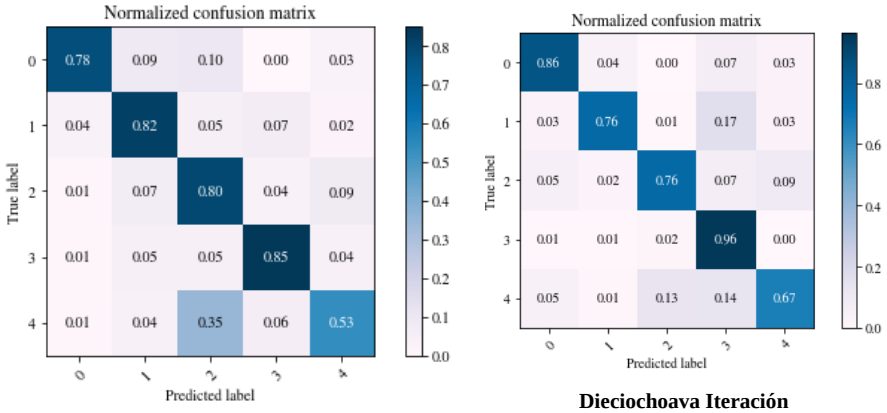
No. Iteración	Arquitectura	Pre – Procesamiento	Profundidad	Parámetros	Regularización	Función de Optimización	No. Épocas	Tiempo prom. Por Época	Exactitud de la última época	Precisión de la última época	Sensibilidad de la última época
13	**Xception**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	36 capas convolucionales + 1 capa densa. Función de activación:ReLU.	20,871,725	No	SGD	100	132 seg	**0.6881**	**0.6898**	**0.6785**
14	**Xception**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	36 capas convolucionales + 1 capa densa. Función de activación:ReLU.	20,871,725	No	SGD	10	138 seg	**0.6922**	**0.7513**	**0.6033**
15	**Xception**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	36 capas convolucionales + 1 capa densa. Función de activación:ReLU.	20,871,725	Yes (Regularización L1 =0.01)	SGD	30	168 seg	**0.5103**	**0.5585**	**0.4569**
16	**Xception**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	36 capas convolucionales + 1 capa densa. Función de activación:ReLU.	20,871,725	Yes (Regularización L1 =0.01)	SGD	100	130 seg	**0.6334**	**0.6617**	**0.6129**
17**	**Xception**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	36 capas convolucionales + 1 capa densa. Función de activación:ReLU.	20,871,725	Yes (Regularización L2 =0.01)	SGD (momentum = 0.1)	50	136 seg	**0.7708**	**0.7784**	**0.7497**
18**	**Xception**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	36 capas convolucionales + 1 capa densa. Función de activación:ReLU.	20,871,725	Yes (Regularización L2 =0.01)	Adam	50	128 seg	**0.803**	**0.8201**	**0.7798**

Nota. Datos de los resultados de las iteraciones 13 a 18.

El mejor resultado se observa con una función de activación ReLU, una función de optimización ADAM, regularización L2 y 100 épocas de entrenamiento (Table 3, Iteración 18). Comparado los resultados presentados hasta el momento, el primer logro importante es que los errores tipos I y II están por debajo del 20% (Figura 31), alcanzando una exactitud del 80%.

Figura 32.

Matrices de confusión - Xception



Diecisieteava Iteración **Dieciochoava Iteración**

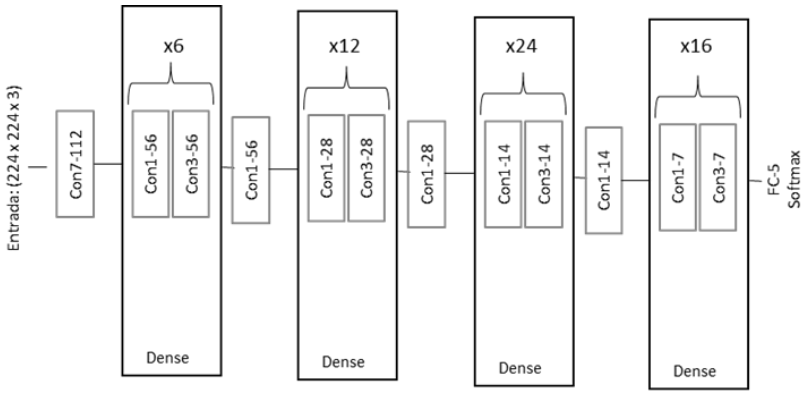
Nota. Datos de los resultados de las iteraciones 17 y 18.

Arquitectura DenseNet-121

Se realiza un último ejercicio con la arquitectura DenseNet (Figura 32), la cual ha mostrado ser substancialmente más profunda (121 capas), más eficaz y eficiente a la hora de hacer el entrenamiento.

Figura 33.

Arquitectura DenseNet-121 utilizada



Nota. Representación gráfica de la red neuronal DenseNet-121 propuesta.

El primer cambio que se analiza es una disminución significativa en el número de parámetros (Tabla 4). Esta arquitectura alcanza una exactitud del 0.78%, siendo aun mayor la alcanzada por la arquitectura Xception.

Tabla 4

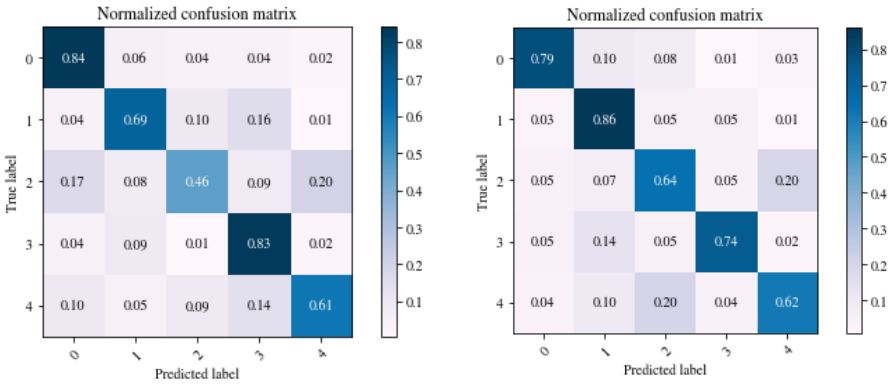
Resultados arquitectura DenseNet

No. Iteración	Arquitectura	Pre – Procesamiento	Profundidad	Parámetros	Regularización	Función de Optimización	No. Épocas	Tiempo prom. por Época	Exactitud de la última época	Precisión de la última época	Sensibilidad de la última época
19	**DenseNet**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	120 Capas Convolucionales + 4 AvgPool + 1 capa densa. Función de activación:ReLU.	7,048,517	No	SGD	50	80 seg.	**0.7045**	**0.743**	**0.6525**
20**	**DenseNet**	**Generación de datos:** rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest', validation_split = 0.2	120 Capas Convolucionales + 4 AvgPool + 1 capa densa. Función de activación:ReLU.	7,048,517	No	SGD	100	80 seg.	**0.7893**	**0.8062**	**0.7798**

Nota. Datos de los resultados de las iteraciones 19 a 20.

Figura 34.

Matrices de confusión - DenseNet



Diecinueveava Iteración **Veinteava Iteración**

Nota. Datos de los resultados de las iteraciones 19 y 20

5. Conclusiones

Se propone usar tres arquitecturas diferentes, para clasificar un conjunto de 3,669 imágenes de flores de cinco clases diferentes. Se demuestra que la arquitectura con mejores resultados es la Xception.

Con una arquitectura Xception de 36 capas convolucionales, regularización L2 de 0.01 y una función de optimización Adam, se alcanzó una exactitud de 0.803, una precisión de 0.8201 y una sensibilidad de 0.7798.

En términos de procesamiento, la arquitectura VGG-16 tomo menos tiempo en entrenar los datos, debido a que la profundidad es menor frente a las otras dos arquitecturas presentadas. Por otro lado, como era de esperarse la arquitectura DeseNet, es la arquitectura con un menor número de parámetros.

Dada la clara similitud que existe entre las clases rosa y tulipán, se propone como trabajo en el futuro hacer uso de alguna técnica de segmentación que ayude a los modelos a lograr hacer la diferenciación entre ambas clases.

Dado que el volumen de datos que se posee en este estudio es bajo, para futuros estudios se propone implementar alguna técnica de transferencia de aprendizaje que reduzca significativamente los tiempos de los entrenamientos de los modelos aquí presentados.

6. Bibliografía

Alex Krizhevsky, I. S. (2012). ImageNet Calssification with Deep Convolutional Neural networks. *NIPS*, 1106-114.

Andreas Kaplan, M. H. (2018). Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *ELSERVIER*.

Benoit Liquet, S. M. (s.f.). *The Mathematical Engineering of Deep Learning*. Obtenido de https://deeplearningmath.org/

Chollet, F. (2017). *Deep Learning with Python.* Shelter Island: Manning Publications Co.

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *Computer Vision Foundation*.

Christian Szegedy, W. L. (2015). Going Deeper with Convolutions. *CVF*, 1-8.

D S Guru, Y. H. (2010). Texture Features and KNN in Classification of Flower Images. *IJCA*, 21-29.

Evaldas Vaiciukynas, A. G. (2018). Parkinson's Disease Detection from Speech Using Convolutional Neural. *Researchgate*.

Gao Huang, Z. L. (2018). Densely Connected Convolutional Networks.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow.* United States: O'Reilly Media, Inc.

Hazem Hiary, H. S. (2018). Flower Classification ussing deep convolutional neural networks. *The Institution of Engineering and Technology*, 855-862.

Honglak Lee, R. G. (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. *Computer Science Department, Stanford University*.

Kaiming He, X. Z. (2016). Deep Residual Learning for Image Recognition. *CVF*.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica* , 249-268.

M.V.D. Prasad, B. J. (2018). An efficient classification of flower images with convolutional neural networks. *International Journal of Engineering & Technology*, 385-391.

Mei Zhang, H. S. (2021). Classification of flower image based on attention mechanism and multi-loss attention network. *El Sevier*, 307-317.

Min Qin, Y. X. (2019). A new Improved Convolutional Neural Network Flower Image Recognition Model. *IEEE*, 3110-3117.

Musab COŞKUN1, Ö. Y. (2017). AN OVERVIEW OF POPULAR DEEP LEARNING METHODS. *European Journal of Technic*.

Nagy, J. Z. (2004). Evaluation of model-based interactive flower recognition. *ICPR 2004.*, 311-314.

Nash, K. O. (2015). An Introduction to Convolutional Neural Networks.

Neha Sharma, V. J. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. *ScienceDirect*, 377-384.

Schank, R. C. (1991). WHERE'S THE AI? *AI Magazine*.

Varma, M. (2007). Learning de discrimiminative power inaviant trade-off. *Journal of Computer Security*, 1- 8.

Xiaoling Xia, C. X. (2017). Inception-v3 for Flower Classification. *IEEE*, 783-787.

Y. LeCun, L. B. (1998). Gradient-Based Learning Applied to Document Recognition. *IEEE*, 2278-2324.

Yuanyuan Liu, F. T. (2016). Flower Classification via Convolutional Neural Network. *IEEE*, 110-116.

Z. Jie, N. G. (2004). Evaluation of model-based interactive flower recognition. *Journal of Computer Society*, 311-314.

Zisserman, K. S. (2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. *ICLR* .

Zisserman, M. N. (2006). A visual vocabulary for flower classification. *CPVR*, 1447-1454.