Red neuronal convolucional para la clasificación de dos especies de roedores

Sebastián Marín-Ruiz *
sebastian.marinr@uqvirtual.edu.co

Resumen

En el presente trabajo se desarrolló una red neuronal convolucional para clasificar dos especies de roedores: guatines y chigüiros, con el objetivo de crear un algoritmo que permita discernir entre una especie y otra. Lo anterior, se puede extrapolar a otros ámbitos tanto de la biología como de la ciencia en general. Mediante la implementación de varias capas convolucionales, de pooling y de capas densas, se estableció una configuración en la cual el modelo entrenado tenía una precisión del 87.75 %, y una precisión de validación del 85.02 %.

1. Introducción

La clasificación de imágenes han sido un problema fundamental en el campo de la visión computacional, ya que describir de manera detallada y específica, por medio de operaciones matemáticas, cómo los seres vivos analizan e interpretan las imágenes que procesa el cerebro, ha suponido un gran desafío para transferirlo a los algoritmos de análisis de imágenes. Por tal motivo, se han ido desarrollando nuevas herramientas y mejorado otras para lograr acercar a los algoritmos cada vez más a la identifiación de patrones y objetos de manera tan desarrollada como lo hacemos los seres humanos. Es así, como se crearon las redes neuronales, las cuales permitían, mediante análisis estadístico, relacionar la información que se le proporcionaba con nuevos datos que se le brindaban y así, clasificarlos.

Sin embargo, esta última tenía la limitación de centrarse únicamente a la información que se le proporcionaba, volviéndose imprecisa cuando los nuevos datos estaban ligeramente modificados de la información con la que había sido entrenada. Fue con la intención de mejorar las redes neuronales, que se desarrolló la red neuronal convolucional, la cual tiene la característica de clasificar mejor las imágenes gracias a su reconocimiento de características, patrones y formas esenciales en un conjunto de imágenes.

Las redes neuronales convolucionales (CNN) han demostrado ser una herramienta poderosa para abordar el desafío de una clasificación de imágenes más precisa. En este escrito, se presenta un proyecto de Física Computacional centrado en el desarrollo de un algoritmo de CNN en Python para clasificar dos grupos distintos de imágenes: guatines y chigüiros.

El objetivo principal de este proyecto es explorar el potencial de las redes neuronales convolucionales en la tarea de clasificación de imágenes, aprovechando su capacidad para aprender características relevantes directamente de los datos. Los guatines y los chigüiros son especies de roedores nativas de nuestra región, y aunque pueden parecer similares a simple vista, presen-

*Programa de Física, Facultad de Ciencias Básicas y Tecnologías - Universidad del Quindío.

tan diferencias en su apariencia física que los distinguen.

El uso de técnicas de aprendizaje automático, como las redes neuronales convolucionales, para identificar y diferenciar imágenes de diferentes categorías tiene aplicaciones prácticas en diversos campos, como la biología, la medicina, la agricultura y la conservación de la fauna, además de poder extrapolar un algoritmo de clasificación de imágenes a la física como clasificación de cuerpos estelares en astronomía, etc. Además, este proyecto brinda la oportunidad de aplicar conceptos y métodos fundamentales de la física computacional en un contexto interdisciplinario, ampliando así nuestras habilidades y conocimientos en el área.

En este informe, se presenta el proceso de desarrollo de un algoritmo de clasificación, incluyendo la adquisición y preparación de los datos de entrenamiento, el diseño de la arquitectura de la red neuronal convolucional, y la evaluación de su rendimiento. También discutiremos los resultados obtenidos y analizaremos las posibles limitaciones y áreas de mejora para futuras investigaciones. Este proyecto representa un paso hacia adelante en la aplicación de la física computacional en problemas de visión computacional y aprendizaje automático.

2. Marco teórico

En esta sección se discuten los conceptos de visión computacional, machine learning y redes neuronales convolucionales, así como la importancia que tienen estas últimas en la clasificación de imágenes.

2.1. Visión computacional

La visión por computadora es una rama de la inteligencia artificial que se centra en la extracción, análisis e interpretación de información visual a partir de imágenes o videos. Su objetivo principal es permitir que las computadoras interpreten y analicen el mundo visual de manera similar a los seres humanos. Lo anterior, se logra mediante una combinación entre las habilidades de programación y estadística, ya que una imagen a nivel computacional es un conjunto de píxeles que se sitúan de forma ordenada. Adicionalmente, si se hace un análisis aún más profundo, cada píxel es un valor numérico que define la tonalidad de cierto color que esté representando. Es decir, una imagen está compuesta de un arreglo de valores numéricos, donde cada uno de estos números indica un píxel con determinadas características.

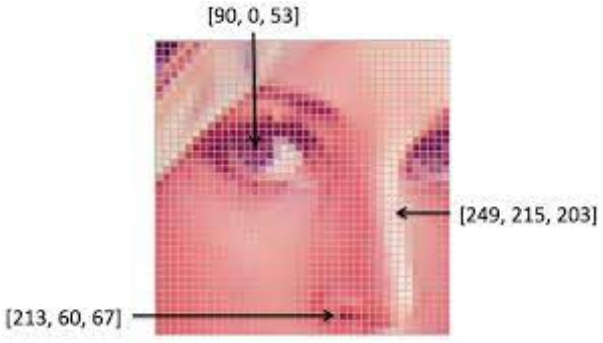

Figura 1: Ejemplo de la interpretación computacional de un pixel [4].

Adicionalmente, cada imagen tiene una dimensión determinada según sea el caso. Las imágenes a color poseen 3 dimensiones, ya que son una superposición de 3 canales de colores primarios: rojo (Red), verde (Green) y azul (Blue), formando así las imágenes RGB. Por lo tanto, una imagen a color está compuesta de una cantidad establecida de pixeles, donde cada pixel tiene 3 valores diferentes, uno para cada color primario, como se puede observar en la figura 1.

2.2. Machine learning

El aprendizaje automático, también conocido como machine learning, es una disciplina de la inteligencia artificial que se ocupa del desarrollo de algoritmos y modelos capaces de aprender patrones y tomar decisiones basadas en datos

sin ser explícitamente programados. Existen diferentes enfoques de machine learning que son necesarios para mejorar la precisión de los modelos predictivos. Dependiendo de la naturaleza del problema que se está atendiendo, existen diferentes enfoques basados en el tipo y volumen de los datos. Algunos de los enfoques son el machine learning supervisado, que se encarga de predecir o interpretar una variable específica, el marchine learning no supervisado, que procesa datos no etiquetados y lo separa mediante técnicas de análisis estadístico o deep learning, que se encarga de interpretar datos no estructurados y hallar un patrón en ellos [3]. En este último, específicamente, hay una técnica novedosa para el procesamiento de imágenes, llamado redes neuronales convolucionales.

2.3. Redes Neuronales Convolucionales

Las redes neuronales convolucionales son un tipo especializado de arquitectura de redes neuronales diseñadas específicamente para el procesamiento de imágenes y datos en forma de cuadrícula, como imágenes en 2D o volúmenes en 3D. Estas redes se basan en la idea de que las características visuales relevantes de una imagen pueden ser aprendidas mediante la aplicación de filtros convolucionales y operaciones de pooling, los cuales reducen la dimensión de la imagen y, a su vez, extraen información de las características más relevantes de una imagen, permitiendo así que el algortimo detecte patrones, formas, etc.

Las capas convolucionales en una red neuronal convolucional se encargan de extraer características locales de la imagen, mientras que las capas completamente conectadas se utilizan para combinar estas características y realizar la clasificación final. El entrenamiento de una CNN implica la optimización de los pesos de las conexiones entre las neuronas, utilizando algoritmos como la retropropagación del error.

2.4. Funciones de Activación

Las funciones de activación desempeñan un papel crucial en las redes neuronales, ya que introducen no linealidades en los cálculos y permiten a la red aprender relaciones complejas entre las características de entrada y las salidas deseadas. Algunas funciones de activación comunes utilizadas en las redes neuronales convolucionales incluyen la función ReLU (Rectified Linear Unit), que desactiva la neurona cuando su valor es menor a cero y es lineal para mayores valores o iguales a este, la función sigmoidal, que es especialmente útil cuando se quiere clasificar un sistema binario y la función softmax.

Finalmente, para evaluar el rendimiento de nuestro algoritmo de clasificación, utilizamos métricas comunes en la visión por computadora, como la precisión (accuracy), la matriz de confusión, la sensibilidad (recall) y la especificidad (specificity). Estas métricas nos permiten cuantificar la efectividad de nuestro modelo y compararlo con otros enfoques o investigaciones relacionadas.

3. Metodología

Para el presente trabajo se utilizaron dos bases de datos: una para los guatines (Dasyprocta punctata) y otra para los chigüiros (Hydrochoerus hydrochaeris). Para estos últimos, se descargó una base de datos de la plataforma Global Biodiversity Information Facility (GBIF), que contenía un archivo tsv con enlaces que dirigían a cada imagen individual [2].

Para los guatines, el autor y una estudiante de biología, Valentina Ríos Giraldo, fueron al Parque de la Vida de la ciudad de Armenia, Quindío, Colombia, y tomaron personalmente las imágenes de dichos animales. En total, se recopilaron 1274 imágenes para cada especie, con el objetivo de que las bases de datos estuvieran balanceadas.

Inicialmente, se tomaron las fotos a los guatines y posteriormente, una por una, se recortaron

para que quedaran guatines individuales en cada imagen y, adicionalmente, que estos ocuparan la totalidad de la misma, para eliminar lo máximo posible el ruido que podría generar el entorno al momento de hacer el procesamiento de las imágenes en el entrenamiento de la red neuronal convolucional (CNN, por sus siglas en inglés). Este procedimiento no se aplicó para ninguna de las especies en la parte de validación del algoritmo.

El mismo proceso se realizó para la base de datos de los chigüiros, con la excepción de que se hizo una depuración de las imágenes descargadas automáticamente, ya que esta base de datos contenía imágenes que no correspondía directamente con el animal en cuestión, sino también imágenes de sus huellas, heces y decesos, razón por la que se eliminaron de manera manual estas imágenes y se conservaron aquellas que mostraran explícitamente a la especie viva. Posteriormente, para cada especie, se dividieron las imágenes: 1000 para la carpeta de entrenamiento (o *train*) y 277 a la carpeta de validación (o *test*).

Luego, se redujo la dimensionalidad de las imágenes a escala de grises, ya que el color de las especies no es un factor relevante en su clasificación, sino sus características morfológicas. Adicionalmente, se hizo un reescalamiento de los datos, con el objetivo de que todas las imágenes utilizadas tuvieran un tamaño de 256x256 píxeles y se aplanaron los datos para posteriormente codificar las etiquetas en cada caso y normalizar la base de datos pre-procesada. Finalmente, se entrenó la red neuronal convolucional con diferentes configuraciones, variando el número de capas, añadiendo y omitiendo dropouts, etc., y se determinó la mejor de ellas para este trabajo.

Los entrenamientos se realizaron con CPU, ya que hubo confusiones al momento de instalar la librería *TensorFlow* en el equipo, razón por la cual no se reconoció la GPU del mismo y los procesos fueron más lentos. Finalmente, en todos los entrenamientos empleados, se usaron 10 épocas.

4. Resultados y discusión

Primero, se realizó un entrenamiento de la red neuronal convolucional, intercalando una capa convolucional y una capa de pooling un par de veces, como se ve en la figura 2.

```
# Definir el modelo
modelo = tf.keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(2, activation='softmax')
])
```

Figura 2: Primera configuración de la red neuronal convolucional con dos capas convolucionales, kernel variable, y una función de activación ReLu.

El entrenamiento de esta red dio una precisión de entrenamiento del 99.5 % y una precisión de validación del 78.9 %, lo cual denota un claro sobreajuste en los datos. Esto, generó la necesidad de ajustar la base de datos o ajustar las capas de la red para mejorar la precisión de validación sin perder la de entrenamiento.

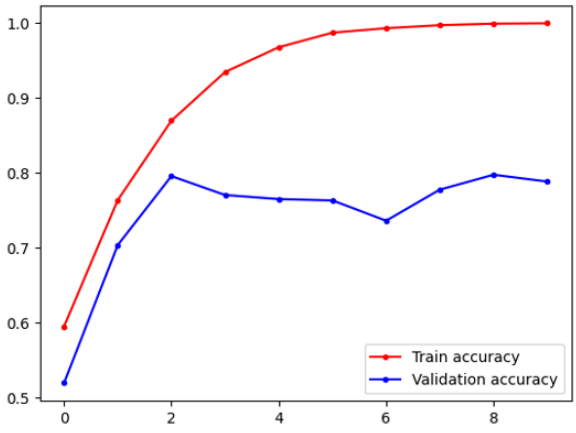

Figura 3: Gráfico de la precisión de entrenamiento y validación, respectivamente para la primera configuración.

```
num_clases = 2

modelo = Sequential()
modelo.add(Conv2D(32, (3, 3), activation='sigmoid', input_shape=(256, 256, 1)))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dropout(0.25))
modelo.add(Conv2D(64, (3, 3), activation='sigmoid'))
modelo.add(MaxPooling2D(pool_size=(2, 2)))

modelo.add(Flatten())
modelo.add(Dense(128, activation='sigmoid'))
modelo.add(Dropout(0.5))

modelo.add(Dense(num_clases, activation='softmax'))
```

Figura 4: Segunda configuración de la CNN. Se cambió la función de activación a sigmoide y se añadió un *dropout*.

Seguidamente, se hizo un segundo entrenamiento en el cual se cambió el tipo de función de activación a sigmoide, ya que este tipo de funciones son más útiles al momento de clasificar valores binarios [1]. Adicionalmente, se introdujeron los *dropouts*, con el objetivo de desactivar neuronas aleatoriamente y reducir el sobreajuste en el entrenamiento, como se puede ver en la figura 4.

Esta configuración de la red dio resultados deficientes, debido a que tuvieron una precisión de entrenamiento de 70.38 % y una precisión de validación del 50.00 %, el cual se mantuvo constante desde la primera época hasta la última.

Finalmente, se añadieron más capas convolucionales que iban duplicando el tamaño del kernel en cada capa de este tipo, una capa de pooling después de cada convolución, y una capa densa de tamaño 128 que iba reduciendo su tamaño a la mitad en cada capa de su tipo. Adicionalmente, se incorporaron *dropouts*, con una probabilidad menor que el anterior (0.2) después de cada proceso en la capa densa, como se puede apreciar en la figura 5.

```
num_clases = 2

modelo = tf.keras.models.Sequential()
modelo.add(Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 1)))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dense(128, activation='relu'))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dropout(0.2))
modelo.add(Conv2D(64, (3, 3), activation='relu', input_shape=(256, 256, 1)))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dense(64, activation='relu'))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dropout(0.2))
modelo.add(Conv2D(128, (3, 3), activation='relu', input_shape=(256, 256, 1)))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dense(32, activation='relu'))
modelo.add(MaxPooling2D(pool_size=(2, 2)))
modelo.add(Dropout(0.2))

modelo.add(Flatten())
modelo.add(Dense(128, activation='relu'))
modelo.add(Dropout(0.5))
```

Figura 5: Tercera configuración de la CNN. Se regresó la función de activación a ReLu y se añadieron más capas, duplicando o diviendo la dimensión de las mismas.

Este último entrenamiento fue el que proporcionó mejores resultados en cuanto a precisión de entrenamiento, con un 87.75 % y de validación, con un 85.02 % por lo que se puede apreciar una disminución del sobreajuste y una mejor predicción de las imágenes de la base de datos, como se puede observar en la figura 6.

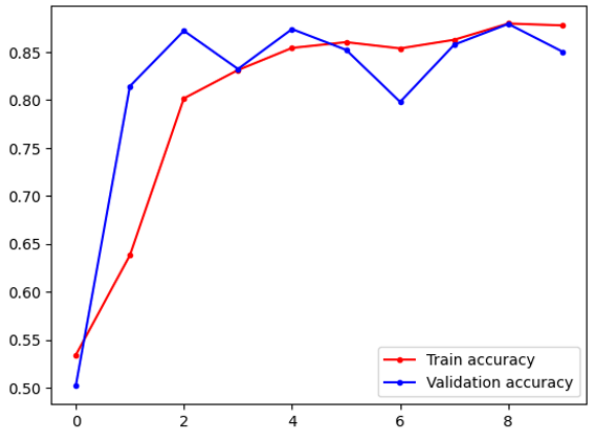

Figura 6: Gráfica de la precisión de entrenamiento y validación, respectivamente, para la tercera configuración.

5. Conclusiones

Como conclusión final, se determinó que la estructura de las redes neuronales convolucionales son la más fundamentada y mejor organizada

de todas las configuraciones. Esto es, aumentando el tamaño del kernel en las capas convolucionales, de tal manera que se duplique en cada conjunto de capas y, a su vez, disminuyendo el tamaño de la capa densa en cada conjunto. Lo anterior, ya que a medida que la imagen va pasando los filtros de la red neuronal, se va disminuyendo su dimensionalidad, razón por la que es pertinente aumentar los kernel de las capas convolucionales para extraer la información más compleja y detallada de los datos iniciales de la misma. Con la estructura mencionada anteriormente, se demostró que se consigue una alta precisión, tanto del modelo entrenado como de la validación del modelo, sin sobreajustes ni pérdidas muy altas.

Bibliografía

[1] Diego Calvo. *Función de activación - redes neuronales*. 2018. URL: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/.

[2] GBIF.Org User. *Occurrence Download*. 2023. DOI: 10.15468/DL.E8AHDQ. URL: https://www.gbif.org/occurrence/download/0263963-230224095556074.

[3] IBM. *¿Qué es Machine Learning?* 2018. URL: https://www.ibm.com/mx-es/analytics/machine-learning.

[4] Raúl E. Lopez Briega. *Visión por computadora*. 2018. URL: https://iaarbook.github.io/vision-por-computadora/.