Evidencia 4. Deep Learning

AUTHORS
Alejandra Velasco Zárate A01635453
José Antonio Juárez Pacheco A00572186
Jose Carlos Yamuni Contreras A01740285
Juan Manuel Hernández Solano A00572208
Mayra Sarahí De Luna Castillo A01635774

PUBLISHED
September 6, 2023

Abstract

Este reporte se enfoca en las aplicaciones de las redes neuronales, más específico, se hace uso de una red neuronal convolucional para el procesamiento y la clasificación de imágenes de perros y gatos. El reporte consta de una introducción donde se explica la importancia en la clasificación de imágenes en el campo del machine learning y la teoría detrás del funcionamiento de las redes neuronales. Para la aplicación y la creación del modelo de clasificación, se toma una muestra de 6,000 imágenes de cada una de las dos especies proporcionadas por Microsoft y se hace uso del framework Tensor Flow para el entrenamiento del modelo. Una vez entrenado el modelo se hace una evaluación de rendimiento a través de una matriz de confusión, un reporte del modelo y se visualiza el progreso de su entrenamiento llevado a cabo a lo largo de las iteraciones de ajuste de pesos y bias. Se cuenta con la gráfica final del rendimiento del modelo tanto de la exactitud como de la pérdida. Al final del trabajo se obtienen suficientes resultados para hacer un análisis de resultados y poder concluir si el uso de redes neuronales convolucionales puede ser útil para la clasificación de imágenes o no, en el caso de este trabajo se obtuvo un modelo aceptable, pero el rendimiento puede mejorar.

Introducción

La clasificación de imágenes es una tarea fundamental en el campo de la visión por computadora y el aprendizaje profundo. La capacidad de diferenciar objetos y reconocer patrones visuales en imágenes ha impulsado una amplia gama de aplicaciones, desde la detección de objetos en automóviles autónomos hasta la identificación de enfermedades en imágenes médicas. En este contexto, el reconocimiento de animales, como perros y gatos, representa un desafío interesante debido a la diversidad de razas, poses y entornos en los que estas mascotas pueden ser fotografiadas.

En este artículo, presentamos un enfoque para la clasificación precisa de perros y gatos mediante el uso de redes neuronales profundas. Nuestra investigación se basa en la utilización de la base de datos libre propiedad de Microsoft, la cual consta de miles de imágenes de gatos y perros. El objetivo principal de este estudio es desarrollar un clasificador de alto rendimiento capaz de distinguir con precisión entre estas dos especies, incluso en situaciones desafiantes.

El aprendizaje profundo, en particular las redes neuronales convolucionales (CNN), ha demostrado ser una herramienta poderosa para la clasificación de imágenes. Aprovechando el poder de estas redes,

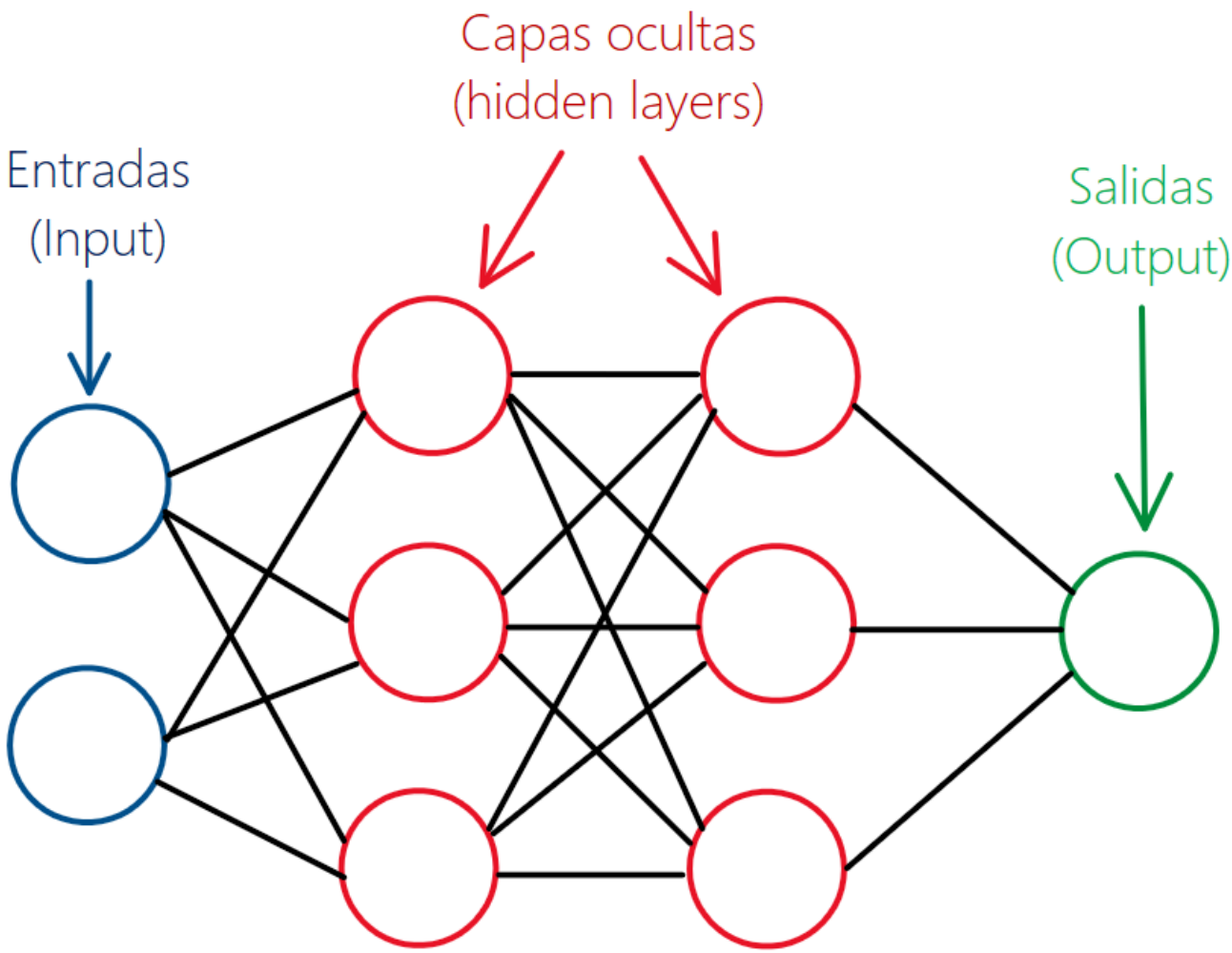
abordamos el problema de clasificar perros y gatos como un ejemplo de clasificación binaria. A medida que avanzamos en este proyecto, exploraremos diferentes arquitecturas de CNN, técnicas de preprocesamiento de datos y estrategias de entrenamiento para lograr el mejor desempeño posible.

Marco Teórico

El aprendizaje profundo es un subconjunto de machine learning, donde las redes neuronales, algoritmos inspirados en cómo funciona el cerebro humano, aprenden de grandes cantidades de datos. Los algoritmos de deep learning realizan una tarea repetitiva que ayuda a mejorar de manera gradual el resultados a través de *deep layers* lo que permite el aprendizaje progresivo. Este proceso forma parte de una familia más amplia de métodos de machine learning basados en redes neuronales (IBM, 2023).

Redes neuronales

Las redes neuronales artificiales (ANNs) son un modelo de un algoritmo computacional inspirado en las redes neuronales biológicas. Son como una herramienta que tiene la capacidad para aprender, generalizar y procesar datos de manera automática, con aplicación en tareas de clasificación y regresión, principalmente en datos con comportamiento complejo y datos no estructurados. En la clasificación se busca organizar los datos de entrada en diferentes clases y en la regresión se busca predecir un parámetro de salida desconocido. A partir de estas aplicaciones, el potencial de las ANNs se encuentra en el reconocimiento de patrones y la predicción de comportamientos(Sarmiento, 2020). La estructura de una red neuronal artifical consiste de nodos, que representan las entradas, las capas ocultas y las salidas, como se observa en la siguiente figura.

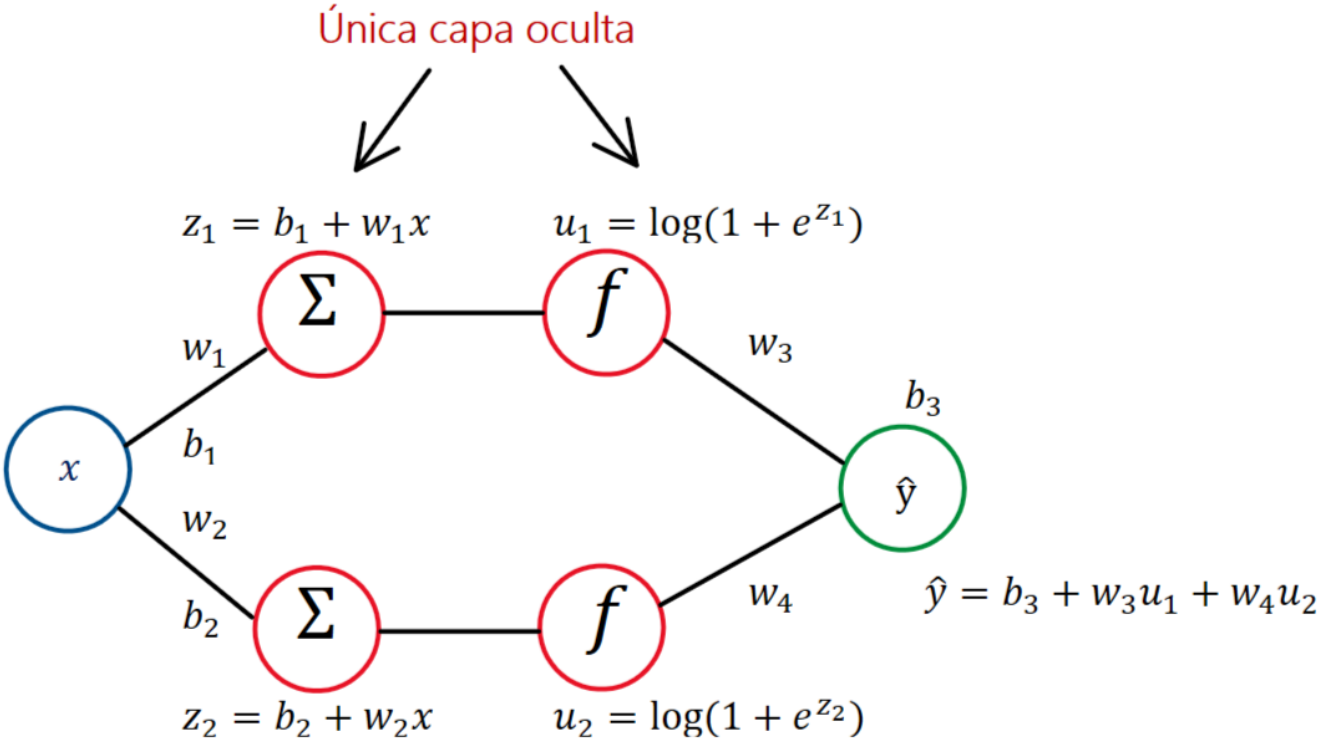


Estructura básica de una red neuronal

En la red neuronal están las múltiples entradas, pesos, biases, suma y funciones de transferencia (activación) y consta de diferentes parámetros. Toda la transformación de los datos sucede dentro de las capas ocultas, que puede ser solo una o más, en ellas los datos se estiran, se encogen y transforman las entradas mediante una función, mejor conocida como función de activación. Existen varios tipos de funciones, las más utilizadas son:

- Softplus : $f(x) = \log(1 + e^x)$
- Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

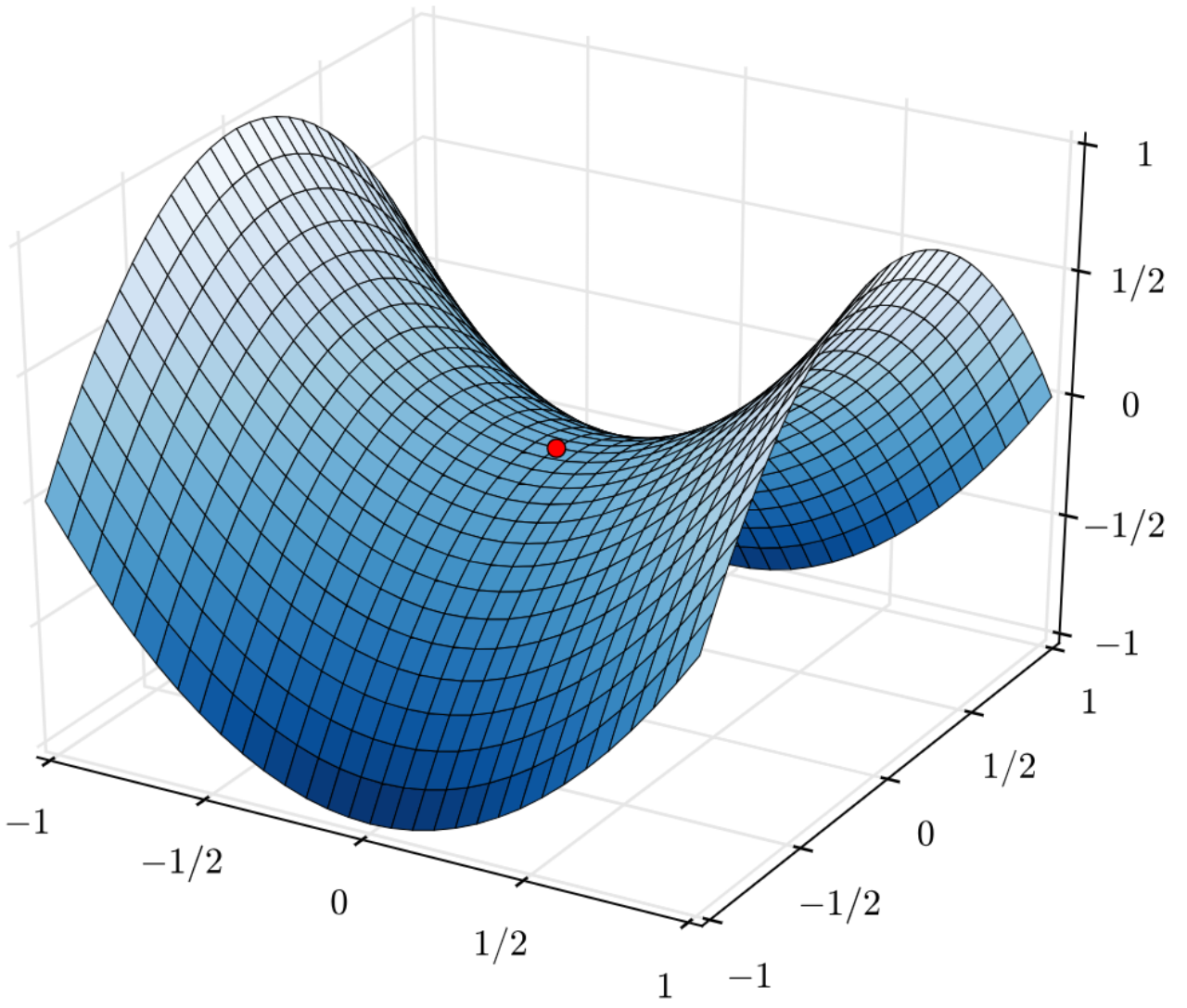
Además, los parámetros de pesos (weights w_i) y biases (b_i) están dentro de los nodos de las capas ocultas y en los nodos de las salidas. Estos parámetros primeramente se suman para obtener una nueva función (z_i), que después entra a la función de activación (softplus, ReLU, sigmoid, u_i). Y al final se tiene la ecuación para predecir las salidas ($\hat{y}$). Con un ejemplo visual, se tiene una sola entrada x en una red de una sola capa y con una sola salida.



Funcionamiento de la red neuronal

donde w_1 y w_2 son los pesos de la capa oculta, b_1 y b_2 son los biases de capa oculta. Estos pesos y biases entrán a la capa oculta donde suceden dos cosas: la suma z_1 y z_2 y la transformación softplus de esas funciones u_1 y u_2. Las funciones de suma consisten de multiplicar la entrada con el peso y sumarlo con el bias: $z_1 = w_1 x + b_1$ y $z_2 = w_2 x + b_2$. Esas funciones entran como la variable dependiente en la función de activación: $f(z_1) = u_1 = \log(1 + e^{z_1})$ y $f(z_2) = u_2 = \log(1 + e^{z_2})$. Terminando los cambios de la entrada x en la única capa oculta, se tienen w_3 y w_4, que son los pesos del nodo de salida y b_3 el bias. Después se obtiene la función para la variable objetivo, que es la que se busca clasificar o predecir: $\hat{y} = b_3 + w_3 u_1 + w_4 u_2$.

El problema es que estos parametros, los pesos y los biases, no son conocidos, por lo que se utiliza backpropagation. Lo que hace este método es minimizar la función de pérdida, error cuadrático: $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$. Esto se logra mediante optimización numérica, ya sea Gradient Descent o cualquiera de sus variaciones. Gradient descent busca encontrar el punto mínimo de una función de perdida $C(\theta)$, como se observa en la figura:



Ejemplo de función de pérdida

Esto lo hace de dos manera: analíticamente o por métodos numéricos. En el primer método requiere que la función tenga una forma cerrada la cual sea derivable. El segundo método es iterativo, en el sentido que comienza en una solución y trata de identificar una serie de mejores soluciones, se usa información local a la posición. La función de perdida $C(\theta)$ con respecto a un vector de parámetros $\theta = (\theta_0, \ldots, \theta_p)$ se minimiza inicializando la función con un valor arbitrario $\theta^{(0)}$ y encontrando su gradiente. Iterativamente se actualiza cada elemento de θ usando:

$$\theta_j^{(k+1)} = \theta_j^{(k)} - \alpha \frac{\partial C(\theta)}{\partial \theta_j}$$

donde α es el learning rate, que escala el gradiente en esa dirección y ajusta la velocidad para llegar al punto mínimo, que multiplica a la parcial de la función de pérdida dada la posición anterior θ_j, mejor conocido como la pendiente. El *stepsize* es la multiplicación de la pendiente por el learning rate. Lo que hace esta función es que a partir de un punto arbitrario, se mueve hacia donde $C(\theta)$ decrece más rápido, hasta llegar a un punto donde converge y los valores de θ se estabilizan.

En el ejemplo propuesto, lo que se busca estimar es $\theta = (b_1, w_1, b_2, w_2, b_3, w_3, w_4)$, la función de pérdida es:

$$C(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

y la función de predicción:

$$\hat{y}_i(\theta) = w_3 z_{1i} + w_4 z_{2i} + b_3$$

Con el bacpropagation se empieza de atrás hacia adelante, por lo que empieza con b_3, para esto se asume que todos los otros parámetros en θ tienen los valores óptimos. Se necesita encontrar:

$$\frac{\partial C(\theta)}{\partial b_3} = \frac{C(\theta)}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial b_3}$$

Los pasos a seguir, que son los pasos que utiliza el bacpropagation en cualquier caso son:

- Establecer $\alpha = 0.1$
- Inicializar $b_3^{(0)} = 0$
- Calcular la derivada de la función de pérdida con respecto a $\hat{y}_i$

$$\frac{\partial C(\theta)}{\partial \hat{y}_i} = \sum_{i=1}^{n}(y_i - \hat{y}_i)$$

Y se calcula el error para cada observación desde $\hat{y_1}$ hasta $\hat{y_n}$. Si $w_3 = -1.22$, $w_4 = 2.3$, $b_2 = 0.57$, $b_1 = -1.43$, $w_2 = -3.53$ y $w_1 = 3.34$, entonces el cálculo para obtener el error es:

$$\begin{aligned}
\hat{y_1} &= b_3 + w_3 u_{11} + w_4 u_{21} \\
&= 0 - 1.22 u_{11} - 2.3 u_{21} \\
&= -1.22 \log(1 + \exp\{-1.43 + 3.34(0)\}) - 2.3 \log(1 + \exp\{0.57 - 3.53(0)\}) \\
&= -2.60
\end{aligned}$$

Entonces:

$$\sum_{i=1}^{3}(y_i - \hat{y}_i) = 7.8$$

Y el nuevo valor de b_3 es:

$$b_3^{(1)} = b_3^{(0)} + 0.1(7.8) = 0 + 0.1(7.8) = 0.78$$

Este proceso se repite hasta que el valores de b_3 no cambie mucho, es decir que converja. Gradient descent se hace para cada parámetro dentro de la red neuronal, por lo que todos se inicializan con un valor ($\theta^{(0)} = (b_1^{(0)}, w_1^{(0)}, b_2^{(0)}, w_2^{(0)}, w_3^{(0)}, w_4^{(0)})$). Para cada parámetro se calcula la derivada parcial de la

función del error $C(\theta)$ con respecto al parámetro que se busca optimizar y se aplica la fórmula de gradient descent hasta que el valor de los parámetros converja.

Así es como funciona una red neuronal que predice o clasifica a groso modo, ya que existen diferentes estructuras y arquitecturas de las ANNs. La topología de la red depende del número de capas ocultas, los nodos por capa oculta y sus las interconexiones. La manera de escoger la arquitectura para un red neuronal es mediante prueba y error, de una manera empírica. Para esto se puede utilizar Cross-validation, que es una técnica para evaluar el rendimiento de un modelo de machine learning dividiendo los datos en múltiples conjuntos, entrenando y evaluando el modelo en cada uno de ellos para obtener una medida más robusta de su capacidad de generalización. En cross validation, se evalúa el desempeño de la ANN y dependiendo de eso se quitan o se añaden hidden layers.

Las ventajas de las redes neuronales es que escoger una arquitectura resulta más simple que escoger un kernel en un algoritmo de Support Vector Machine o una base de funciones para ajustar datos con relaciones no lineales, pueden tomar información de los datos vecinos para extraer más información (información espacial) y ofrecen un desempeño superior cuando existen muchas características.

Red neuronal convolucional

Una de las mayores aplicaciones en las redes neuronales, es la clasificación de datos no estructurados como imágenes. Las redes neuronales convolucionales CNN se utilizan para tareas de clasificación y visión artificial, como clasificar imágenes y reconocer objetos. Las CNN se distinguen de otras redes neuronales por su mejor desempeño con entradas de señal de imagen, voz o audio. Tiene tres tipos de capas:

- Capa convolucional
- Capa de agrupamiento
- Capa totalmente conectada (FC)

La capa convolucional es la primera capa de una red convolucional. Si bien a las capas convolucionales las pueden seguir capas convolucionales adicionales o capas agrupadas, la capa totalmente conectada es la capa final. Con cada capa, la CNN aumenta su complejidad, identificando mayores porciones de la imagen. Las primeras capas se enfocan en características simples, como colores y bordes. A medida que los datos de la imagen avanzan a través de las capas de CNN, se comienzan a reconocer elementos o formas más grandes del objeto, hasta que finalmente se identifica el objeto previsto (IBM, 2023).

Implementando un preprocesamiento a la imagen para capturar su estructura espacial, se puede aplicar una convolución a la imagen con un kernel. Este filtro (kernel) es una matriz de dimensión fija moviéndose sobre la imagen que resume su información con un solo número. Después de aplicar la convolución, la imagen se comprime más usado un *pooling*, que es otra matriz que extrae aún más información después de la convolución. El pooling más popular es el 'Max-pool' que toma el valor más grande de una región. La forma en la que se obtiene la matriz kernel es que el modelo aprende los números óptimos para capturar información clave dentro de una imagen no directamente interpretable

por humanos. Después de la convolución y el pooling se obtiene un *feature map*, que es una matriz de menor dimensión resultante. Se aplica una y otra vez hasta que se tenga una imagen muy reducida y los elementos de esta matriz son las entradas a una red ANN como las que se explicaron anteriormente. Se aprende esta ANN, usando generalmente ReLU como función de activación y backpropagation. Al final esta red devuelve un valor numérico y a partir de un umbral se decide la categoría a la que pertenece.

Así como las ANNs y las CNNs existen muchos otros tipos de redes neuronales con aplicaciones diferentes y arquitecturas diferentes, pero todas son de mucho valor debido a su capacidad para aprender y generalizar a partir de datos, lo que les permite abordar una amplia gama de problemas complejos en campos como la visión por computadora, el procesamiento de lenguaje natural, la toma de decisiones autónomas y muchas otras aplicaciones.

Metodología y aplicación

Datos del proyecto

Para este proyecto de clasificación se descargó la base de datos de Microsoft que contiene alrededor de 25,000 imágenes de perros y gatos. Al tener una base de datos muy grande, el procesamiento de las imágenes tendría un alto costo computacional, es por ello que únicamente se eligen 6,000 muestras para las imágenes de perros y 6,000 muestras para las imágenes de gatos. Para asegurar que las imágenes se descargaron de manera correcta se imprime una imágen de la base de perros y una imágen de la base gato. Se realiza una categorización con “labels”: 0 en caso de ser gato y 1 en caso de ser perro.

(a) Gato. (b) Perro.

Conjunto de datos

Teniendo las imágenes cargadas, se establece un formato del tamaño, en este caso 128 píxeles de ancho y 128 píxeles de alto, las imágenes redimensionadas se almacenan en la lista x y las etiquetas correspondientes en la lista y. Debido a que estamos generando una red neuronal, debemos dividir las imágenes en conjunto de entrenamiento que servirá para generar el modelo, y conjunto de prueba, con el que se podrá evaluar su rendimiento y precisión. Se realiza un submuestreo estratificado para equilibrar las clases en el conjunto de entrenamiento con el objetivo de mejorar el rendimiento al reducir el impacto de la clase mayoritaria en el entrenamiento, esto permite que el modelo aprenda de manera más equitativa de todas las clases disponibles.

Construcción de la CNN

Teniendo ya las clases equilibradas en el conjunto de entrenamiento, se procede a construir la arquitectura de la red neuronal artificial (ANN). Se define el número de canales de color en las imágenes de entrada, en este caso se trabaja con RGB (rojo, verde y azul), se crea un modelo de red neuronal secuencial, en donde las capas se organizan una después de la otra sin formar conexiones complejas o ramificaciones entre capas. Esta estructura es la ideal para el procesamiento de datos estructurados como las imágenes.

El modelo consta de varias capas en las que se incluyen convolución, normalización, pooling (agrupación), abandono (dropout) y capas de conexión. Se llevó a cabo tres veces la capa de convolución en la que se aplicaron filtros a las imágenes de entrada para extraer características importantes y reducir la dimensionalidad con la función de activación “ReLu”. En cada una de estas capas se realizaba una normalización de los datos para mejorar la estabilidad , un maxpooling para comprimir más la imágen y resaltar características importantes y un dropout para prevenir el sobreajuste del modelo. Finalmente se aplanan las salida de las capas anteriores y se produce la salida final de la red neuronal, en donde el modelo se compila utilizando la función de pérdida de entropía cruzada binaria y el optimizador “adam”. Como salida de esta red neuronal, obtenemos 12940801 parámetros entrenables, los cuales permiten que la red aprenda de los datos y haga predicciones precisas.

```
=================================================================
Total params: 12942273 (49.37 MB)
Trainable params: 12940801 (49.37 MB)
Non-trainable params: 1472 (5.75 KB)
_________________________________________________________________
```

Parámetros

```
Epoch 1/10
274/274 [==============================] - 260s 949ms/step - loss: 0.2313 -
accuracy: 0.9049
Epoch 2/10
274/274 [==============================] - 272s 994ms/step - loss: 0.2378 -
accuracy: 0.9009
Epoch 3/10
274/274 [==============================] - 272s 993ms/step - loss: 0.2065 -
accuracy: 0.9171
Epoch 4/10
274/274 [==============================] - 269s 983ms/step - loss: 0.1803 -
accuracy: 0.9313
Epoch 5/10
274/274 [==============================] - 12617s 46s/step - loss: 0.1559 -
accuracy: 0.9428
Epoch 6/10
274/274 [==============================] - 270s 986ms/step - loss: 0.1798 -
accuracy: 0.9278
Epoch 7/10
274/274 [==============================] - 271s 989ms/step - loss: 0.1538 -
accuracy: 0.9402
Epoch 8/10
274/274 [==============================] - 271s 989ms/step - loss: 0.1085 -
accuracy: 0.9592
Epoch 9/10
274/274 [==============================] - 271s 990ms/step - loss: 0.0992 -
accuracy: 0.9622
Epoch 10/10
274/274 [==============================] - 271s 990ms/step - loss: 0.1189 -
accuracy: 0.9548
```

Epochs

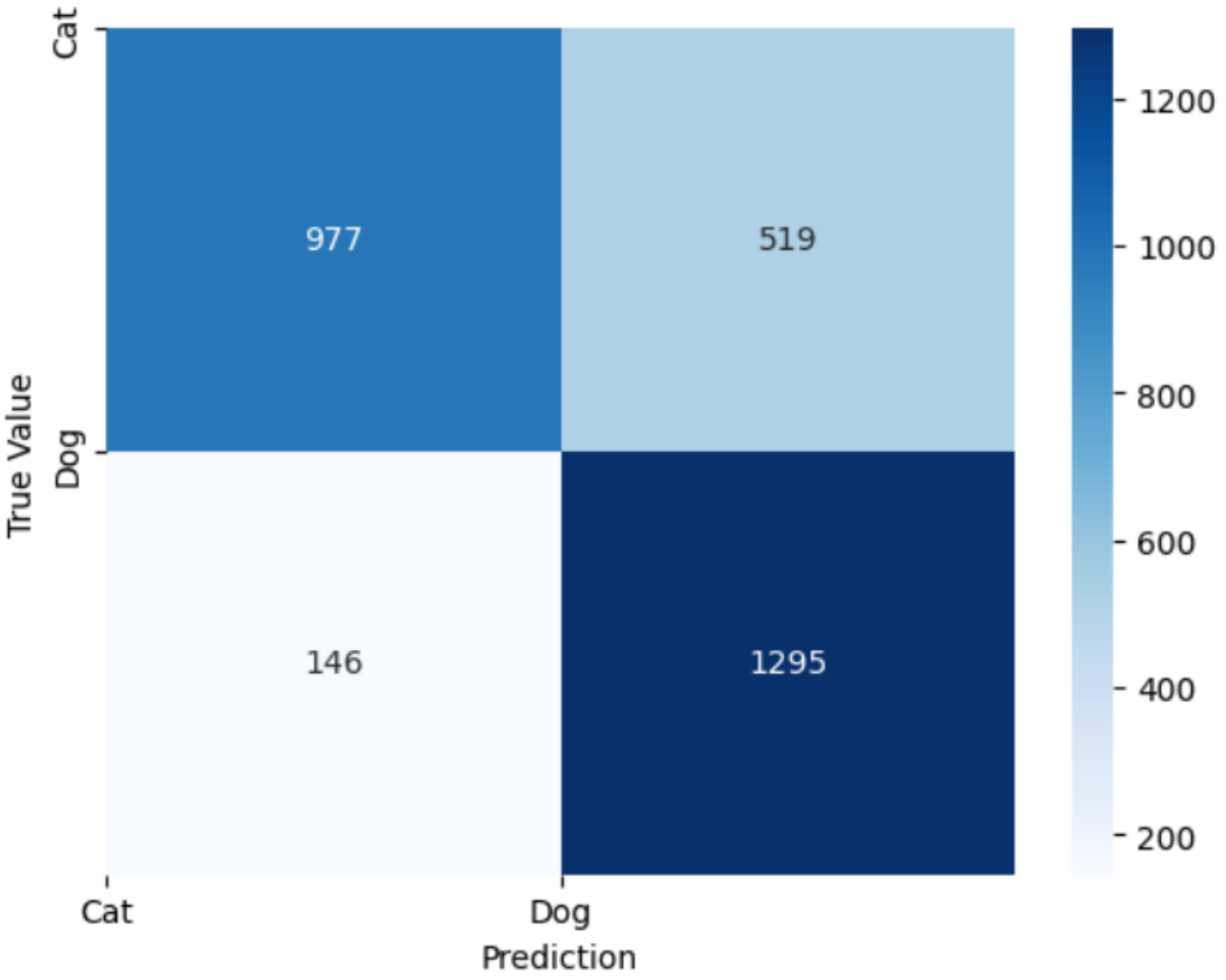
El modelo se entrena con el “train set” equilibrados y utilizando 10 epochs, es decir, los datos pasarán a través del modelo 10 veces, en donde en cada corrida el modelo habrá visto y ajustado sus pesos y sesgos en función de todos los datos de entrenamiento. En cada epoch el modelo arroja una exactitud mayor a 0.90 y termina las corridas con una exactitud de 0.9548, lo que significa que es un gran modelo con los datos de entrenamiento.

Métricas del modelo

	precision	recall	f1-score	support
0	0.87	0.65	0.75	1496
1	0.71	0.90	0.80	1441
accuracy			0.77	2937
macro avg	0.79	0.78	0.77	2937
weighted avg	0.79	0.77	0.77	2937

Métricas de evaluación del modelo

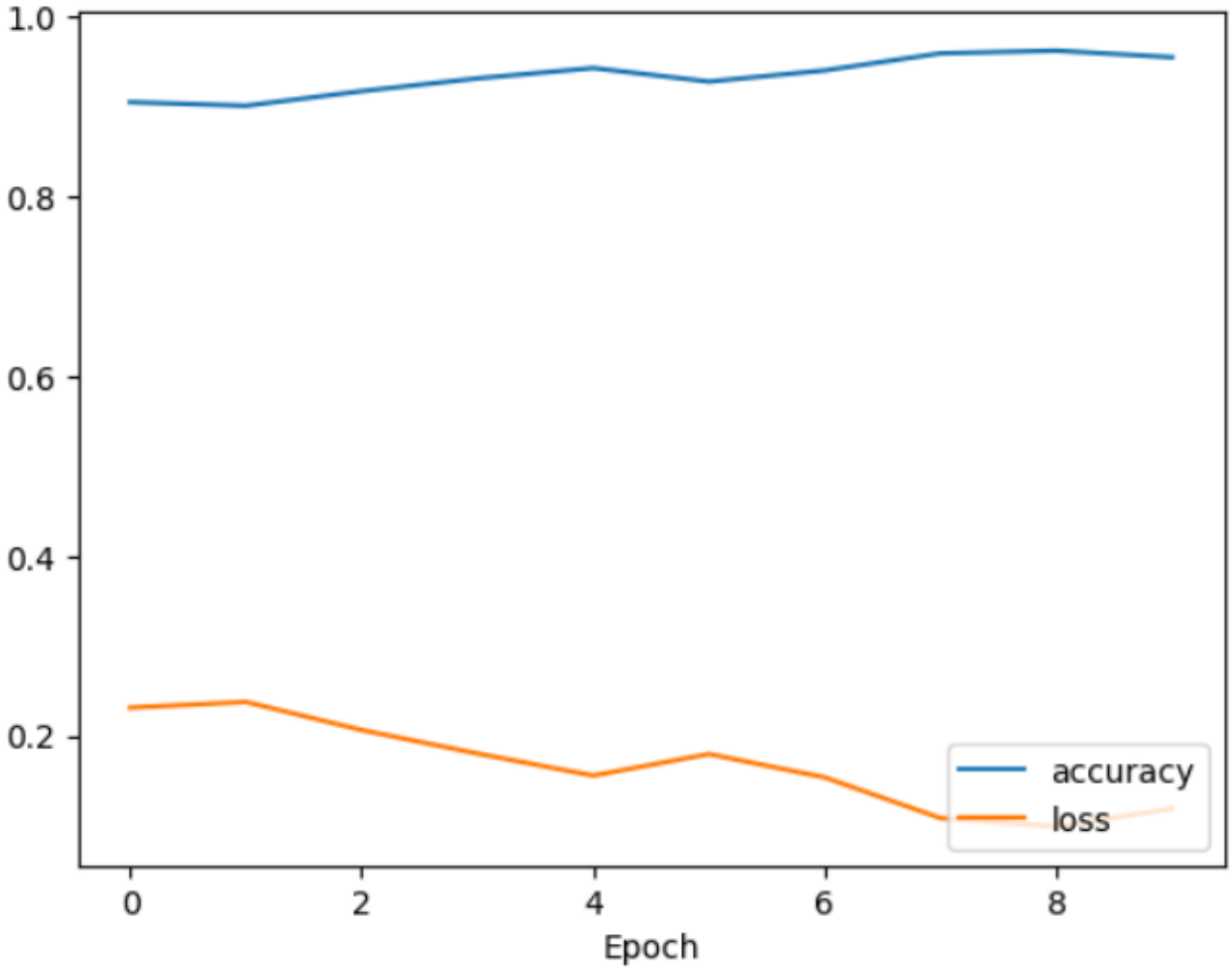
Al momento de comparar el modelo con los datos de prueba nos arroja una exactitud de 0.7736, lo cuál es un modelo aceptable que tiene áreas de mejora.



Matriz de Correlación del modelo

Con la matriz de confusión, se puede observar el problema con el recall que se mencionó anteriormente. El modelo evaluó correctamente a los gatos 977 veces, a los perros 1295, pero tuvo 519 falsos positivos (gatos que fueron clasificados como perros) y 146 falsos negativos (perros que fueron clasificados como gatos).

La gráfica de *accuracy vs loss* nos muestra que mientras más "epochs" se realicen en la red neuronal, la precisión aumenta y la pérdida disminuye, lo que nos indica que el modelo tiene un buen aprendizaje de los datos de entrenamiento.



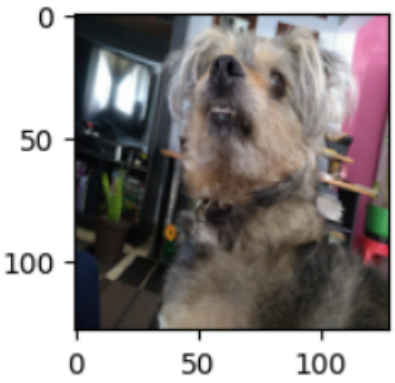
Gráfica de Accuracy vs Loss

Prueba del modelo

Finalmente, el modelo se puso a prueba con imágenes de las mascotas de los integrantes del equipo, del profesor y de Internet. Todas las imágenes se clasificaron correctamente , y para explorar el comportamiento de la red, pusimos una imagen de un hámster, la cuál la clasifica como un gato.

Prueba 1: Maya (perro)

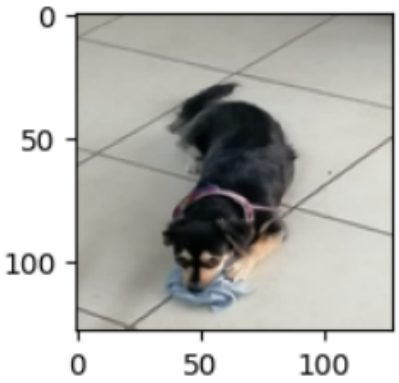
```
1/1 [==============================] - 0s 78ms/step
[[0.90240574]]
class: 1 name= dog
```



Prueba 1: Maya

Prueba 2: Kira (perro)

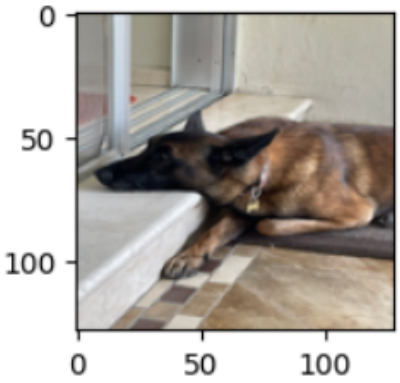
```
1/1 [==============================] - 0s 71ms/step
[[0.88084835]]
class: 1 name= dog
```



Prueba 2: Kira

Prueba 3: Bruce (perro)

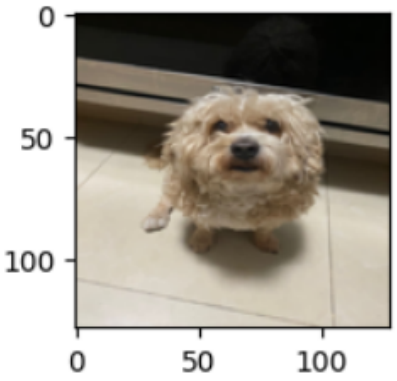
```
1/1 [==============================] - 0s 59ms/step
[[0.8047513]]
class: 1 name= dog
```



Prueba 3: Bruce

Prueba 4: Bubaldihno (perro)

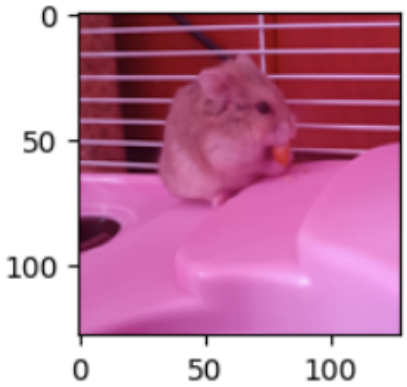
```
class: 1 name= dog

C:\Users\Usuario\AppData\Local\Temp\ipykernel_2724\3973247619.py:3:
DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10
(2023-07-01). Use LANCZOS or Resampling.LANCZOS instead.
  fileImage = Image.open("bubaldo.jpeg").convert("RGB").resize([IMAGE_WIDTH,IMAG
E_HEIGHT],Image.ANTIALIAS)
```



Prueba 4: Bubaldihno

Prueba 5: Burbuja (hámster)

```
1/1 [==============================] - 0s 65ms/step
[[0.00516337]]
class: 0 name= cat
```



Prueba 5: Burbuja

Resultados

Al principio se pensaba utilizar 6000 imágenes de perros y 6000 imágenes de gatos, pero el modelo estaba desbalanceado al momento de separar el conjunto de entrenamiento y de prueba, por lo que se clasificaba mal a los gatos. Es por eso que se hizo un submuestreo para asegurar que la cantidad de imágenes de gatos y de perros estén balanceadas en ambos conjuntos. Esto ayuda a mejorar la precisión por clase y la exactitud por clase. Al hacer el submuestreo quedaron 11685 imágenes en total, que se separaron en conjunto de prueba y entrenamiento y después se utilizaron para construir la red neuronal convolucional y entrenarla con 10 epochs, aplicando convolución y pooling varias veces para reducir la imagen. Gracias a eso se pudo obtener un reporte de clasificación, donde en la categoría de gatos (0) se obtuvo una precisión de 87, lo cual es un buen resultado y aceptable en un modelo. El recall de la clase de los gatos es de 0.65 y en este caso no es un resultado favorable, lo mínimo aceptable sería un recall por clase arriba de 0.70 y el f1-score tiene un valor aceptable de 0.75, por lo que en general las métricas de evalución de la clase gatos es buena. Por otra parte, la clase de perros (1) tuvo una precisión de 71, lo cual es baja pero es aceptable para aplicar este modelo, el recall de esta clase es de 0.90 y es mucho mejor que el recall por clase de los gatos. Claramente este es un buen valor, sumado con el f1-score de la clase de los perros que fue de 0.80, las métricas de la clase 1 son muy buenas, siendo este modelo muy bueno para clasificar perros. En general, la exactitud de este modelo es de 0.77, entre más cercano a 1 es mejor, pero un valor arriba de 0.75 es aceptable, ya que en datos estocásticos y no estructurados es practicamente imposible tener una exactitud muy cercana a 1. Al momento de evaluar el modelo con las imágenes de las mascotas de los integrantes del equipo, la red neuronal convolucional clasificó todas las fotos correctamente, con esto podemos concluir que el modelo es buen modelo gracias a su rendimiento al momento de clasificar, se conoce que no es perfecto y que puede haber casos donde no se clasifique bien la imagen pero en general es un modelo aplicable.

Conclusión

El clasificador tuvo resultados aceptables ya que su rendimiento general al momento de clasificar es bueno, sin embargo la sensibilidad de la clase de los gatos estuvo baja, es decir, que la red neuronal convolucional en algunas ocasiones tiende a clasificar a los gatos como perros. Si se aumenta el número de “epochs” con las que se entrenó el modelo esta sensibilidad por clase pudiera ser más alta, lo cual podría conllevar a un modelo con pesos y bias mejor ajustados al conjunto de datos. Realizar lo último trae como consecuencia un consumo extremadamente alto de recursos computacionales, por lo que no es viable aumentar el número de “epochs” para mejorar el rendimiento del modelo. Al final, observando el comportamiento general del modelo y su exactitud al momento de catalogar diferentes imágenes. Para este proyecto de clasificación se descargó la base de datos de Microsoft que contiene alrededor de 25,000 imágenes de perros y gatos. Al tener una base de datos muy grande, el procesamiento de las imágenes tendría un costo alto computacional, es por ello que únicamente se eligen 6,000 muestras para las imágenes de perros y 6,000 muestras para las imágenes de gatos. Se puede concluir que las redes neuronales convolucionales pueden llevar a ser muy útiles para el procesamiento y clasificación de imágenes. En datos obtenidos de eventos estocásticos de la vida diaria, las redes neuronales pueden ser muy poderosas y muy buenas porque pueden aprender y adaptarse a patrones complejos en datos, lo que las hace efectivas en una amplia variedad de aplicaciones, desde reconocimiento de voz hasta clasificación de imágenes.

Referencias

¿Qué son las redes neuronales convolucionales? (2023). *IBM*. Recuperado de https://www.ibm.com/mx-es/topics/convolutional-neural-networks

Deep Learning. (2023). *IBM*. Recuperado de https://www.ibm.com/mx-es/cloud/deep-learning

Garrido, J. (2023). Deep Learning. [PDF].

Microsoft. (2023, 15 agosto). Download Kaggle Cats and Dogs Dataset from official Microsoft Download Center. Microsoft Store - Download Center. https://www.microsoft.com/en-US/download/details.aspx?id=54765

Sarmiento, J. L. (junio 2020). Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica. *Revista UIS Ingenierías*, 4. Recuperado https://www.redalyc.org/journal/5537/553768213002/html/