Clasificación de Mamíferos de la orden Roentia desde Imágenes Usando Redes Convolucionales
Temas Avanzados en Computación

Por Alexis Susanibar, César Rico y Leandro Lazo

1. Introducción

Los mamíferos pequeños no solo habitan en zonas de vegetación como bosques o parques, sino también en zonas urbanas. Esto llega a tal punto que algunas especies son criadas como mascotas en distintos hogares, ya sea grandes o pequeños como apartamentos.

Así como estos son cuidados dentro de su hábitat, para los cuales se requiere la debida identificación de cada especie, también deben de ser controlados en zonas urbanas debido a que son una fuente de enfermedades a través de sus desechos o por mordedura de estas. Inclusive algunos de los mamíferos que inicialmente fueron adiestrados como mascotas terminan perdidos en la calle contagiándose de las enfermedades que pertenecieron inicialmente a animales silvestres.

Con el avance de las tecnologías del aprendizaje de máquina, sobre todo el campo de las redes neuronales convolucionales, se ha implementado una gran cantidad de usos en el reconocimiento de grupos de fauna, ya sea por imágenes o en captura de video. Ejemplos de estos tenemos, la clasificación de aves, identificación de sonidos de mamíferos marinos, diferenciar de forma automatizada roedores de aves, entre otros.

Una de las mayores complicaciones con respecto a la clasificación de especies es la existencia de una gran diversidad de las mismas y sus respectivas subespecies. De esta manera encontrar las diferencias entre las opciones existentes resulta desafiante por las similitudes en la apariencia que existen entre estas. Por ejemplo, en la Figura 1 se puede observar a una musaraña y en la Figura 2 un pequeño ratón. Se puede observar que las diferencias no son significativas y para una persona no familiarizada con el tema podría ser complicado definir a qué especie pertenece cada uno.

Figura 1: Musaraña

Figura 2: Ratón pequeño

Por estas razones la clasificación de imágenes de animales es muy amplia, ya que permite clasificar las diferentes especies que se encuentran. Para poder realizar esto se aplican las redes neuronales en forma de las redes convolucionales, alcanzando resultados con predicciones de 90% de exactitud aproximadamente.

Para este trabajo de investigación se decidió usar las redes convolucionales variando las capas, optimizadores o reguladores, de tal manera que se consigan resultados óptimos sin tener que incluir tiempos de entrenamiento muy elevados.

El presente trabajo se dividirá en el estado del arte, metodología propuesta, experimentación y resultados, discusión y las conclusiones que se obtuvieron después de ejecutar el modelo propuesto.

2. Estado del arte

Como se mencionó anteriormente las aplicaciones de animales usando redes neuronales convolucionales es amplio. En Nueva Zelanda muchos mamíferos pequeños son considerados pestes, por lo que se aplicó las redes convolucionales para diferenciarlos de otros animales en este caso aves usando descriptores de Fourier y YOLO [1].

Por otra parte también se usa para reconocer el sonido de mamíferos marinos, como es el caso de Gaetz et al que identifica orcas solitarias en base a los sonidos que genera [3].

3. Metodología

Como base usamos imágenes de CIFAR-100, el cual contiene varios conjuntos de clases entre ellos mamíferos pequeños: conejos(rabbit), musarañas(shrew), hamsters, ardillas(squirrel), ratones(mouse). De esta manera obtendremos un modelo inicial por donde se puede llegar a mejorar mediante diferentes parámetros.

Para la parte de preprocesamiento se separan las imágenes objetivo mencionadas anteriormente del resto del dataset con sus respectivos labels. Sin embargo, los labels mantenían el código original del dataset, por lo que se hizo un one hot encoder para dar un nuevo label a cada clase que tendría como valores numéricos del 0 al 4.

Otra cosa que tener en cuenta es la calidad de las imágenes que son de 32 x 32. Se implementó dos modelos para el entrenamiento, el primero constaba de 3 capas convolucionales la cuales tenían 16, 32 y 64 filtros respectivamente, además a cada capa se le aplicaba una activación rectificador linear (ReLu). Luego se redimensiona el resultado con el flatten y finalmente el resultado termina en por dos capas densas adicionales, donde la última es el output con una función de activación softmax debido a que se está clasificando más de dos clases.

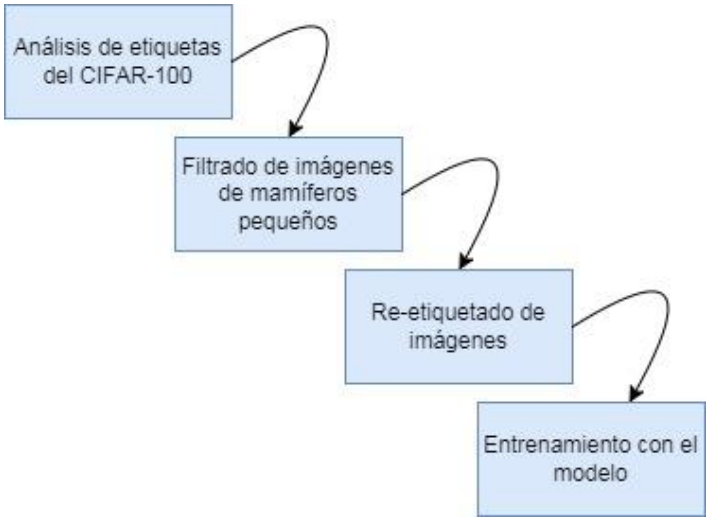
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 3)]       0
conv2d (Conv2D)              (None, 32, 32, 16)        448
re_lu (ReLU)                 (None, 32, 32, 16)        0
conv2d_1 (Conv2D)            (None, 32, 32, 32)        4640
re_lu_1 (ReLU)               (None, 32, 32, 32)        0
conv2d_2 (Conv2D)            (None, 30, 30, 64)        18496
re_lu_2 (ReLU)               (None, 30, 30, 64)        0
flatten (Flatten)            (None, 57600)             0
dense (Dense)                (None, 256)               14745856
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 14,772,010
Trainable params: 14,772,010
Non-trainable params: 0
_________________________________________________________________
```

Figura 3: Resumen de las capas del modelo

Una vez obtenidas las imágenes a procesar se obtuvo 500 imágenes por cada clase, llegando a un total de 2500 en total. Se decidió como estrategia de validación, usar 100 imágenes aleatorias de cada clase para la validación y el resto para el entrenamiento.

Por último se escogió como tamaño de batch 100 debido a que las imágenes no eran demasiadas pesadas de procesar y 100 ciclos o épocas de entrenamiento, con un early stopping de 50 ciclos de paciencia. Aproximadamente cada ciclo con estos parámetros demoraba 2 segundos y se generó un sobreentrenamiento a partir del ciclo 80.

Finalmente la siguiente gráfica explica el proceso descrito previamente.



[1] S. Unger, "Identification of Orcinus orca by underwater acoustics in Dabob Bay," *Oceans '97. MTS/IEEE Conference Proceedings*, 1997, pp. 333-338 vol.1, doi: 10.1109/OCEANS.1997.634385.

[2] K. Shim, A. Barczak, N. Reyes and N. Ahmed, "Small mammals and bird detection using IoT devices," *2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2021, pp. 1-6, doi: 10.1109/IVCNZ54163.2021.9653430.

4. Métrica de evaluación

El modelo propuesto en este trabajo de investigación usará el accuracy, ya que esta métrica se adapta de manera correcta a lo señalado en el trabajo. Además, se puede considerar una de las métricas más usadas en el ámbito de clasificación de imágenes. Su fórmula es la siguiente:

Accuracy = NP / NPT

NP: Número de predicciones acertadas

NPT: Número de predicciones totales.

Asimismo, el conjunto de datos empleado en este trabajo es equilibrado y no cuenta con una cantidad mayor de las clases de un tipo. De esta forma se busca tener información correcta y acertada para el modelo.

5. Experimentación y Resultados

5.1 Descripción del conjunto de datos

Como se mencionó anteriormente se usó como base el dataset de CIFAR-100, el cual tiene 500 imágenes por cada variedad de mamíferos pequeños en tamaño de 32 x 32 píxeles, entre ellos hamsters, ratones, conejos, musarañas y ardillas.

Así también se decidió probar el modelo con otro dataset similar, el CIFAR-10 contiene imágenes de tamaño de —-, con el fin de verificar el desenvolvimiento del modelo desarrollado.

Por último se usó la herramienta Selenium para

realizar un scrapping de imágenes de los mamíferos a estudiar. Llegando a tener un dataset de 100 imágenes aproximadamente de diferentes tamaños entre aproximadamente 250 píxeles y 3500 de ancho y altura.

5.2 Entorno de experimentación empleado

Se utilizó el entorno de Google Colaboratory. Esto se debe a que ofrece un servicio de préstamo de recursos para la ejecución de notebooks y entrenamiento de modelos de redes neuronales. Normalmente el servicio ofrece GPU, sin embargo en muchas ocasiones el servicio llegó a bloquear el acceso a este cuando se usa de manera constante en un corto periodo de tiempo.

5.3 Reporte de entrenamiento

Se realizaron diversos entrenamientos para la clasificación de mamíferos pequeños. El primero usó el dataset de CIFAR-100, extrayendo las imágenes necesarias de los animales de interés. Se usó el optimizador Adam, función de pérdida sparse categorical cross entropy y se configuró los siguientes parámetros:

batch_size=32

Al momento de ejecutar el entrenamiento, se generó un error debido a que las imágenes conservan sus etiquetas originales. Por ende, se realizó un one hot encoder para estandarizar las etiquetas de las imágenes a las salidas del modelo.

El modelo se desarrolló con 2 392 101 parámetros como se muestra en la figura 5 y el entrenamiento género las gráficas de pérdida y accuracy mostrada en la figura 4. Se puede apreciar que el accuracy logrado llega a superar el 60% siendo el máximo 66.60%

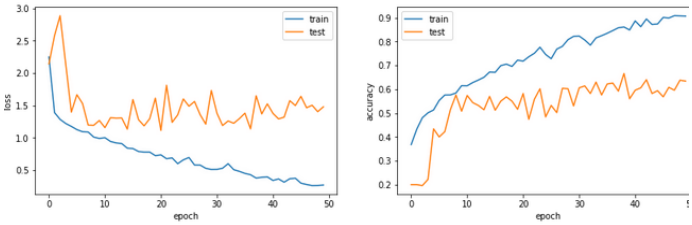


Figura 4: Gráficas de pérdida y accuracy del modelo 1

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 3)]       0

conv2d (Conv2D)              (None, 32, 32, 32)        896

batch_normalization (BatchN  (None, 32, 32, 32)        128
ormalization)

conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248

batch_normalization_1 (Batc  (None, 32, 32, 32)        128
hNormalization)

max_pooling2d (MaxPooling2D  (None, 16, 16, 32)        0
)

conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496

batch_normalization_2 (Batc  (None, 16, 16, 64)        256
hNormalization)

conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928

batch_normalization_3 (Batc  (None, 16, 16, 64)        256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)          0
2D)

conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856

batch_normalization_4 (Batc  (None, 8, 8, 128)         512
hNormalization)

conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584

batch_normalization_5 (Batc  (None, 8, 8, 128)         512
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)         0
2D)

flatten (Flatten)            (None, 2048)              0

dropout (Dropout)            (None, 2048)              0

dense (Dense)                (None, 1024)              2098176

dropout_1 (Dropout)          (None, 1024)              0

dense_1 (Dense)              (None, 5)                 5125

=================================================================
Total params: 2,392,101
Trainable params: 2,391,205
```

Figura 5: Resumen de las capas del modelo 1

El siguiente cambio significativo que se realizó, fue usar transfer learning con resnet50, como se aprecia en la figura 6. La cantidad de parámetros aumentó considerablemente a 23 620 666. Así también, se implementó fine-tuning para el congelamiento de algunas capas del modelo llegando a tener parámetros del modelo que no pueden ser entrenados.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 resnet50 (Functional)       (None, None, None, 2048)  23587712

 max_pooling2d (MaxPooling2D  (None, 1, 1, 2048)       0
 )

 dropout (Dropout)           (None, 1, 1, 2048)        0

 flatten (Flatten)           (None, 2048)              0

 dense_2 (Dense)             (None, 16)                32784

 dense_3 (Dense)             (None, 10)                170

=================================================================
Total params: 23,620,666
Trainable params: 23,567,546
Non-trainable params: 53,120
_________________________________________________________________
```

Figura 6: Resumen de las capas del modelo 2

De este modelo se obtuvieron los siguientes resultados. El accuracy no llegó a alcanzar el 60% obteniendo como máximo 52.60% y gracias al Early Stopping el entrenamiento se detuvo en la época 23 debido a sobreentrenamiento como se puede apreciar en la figura 7.

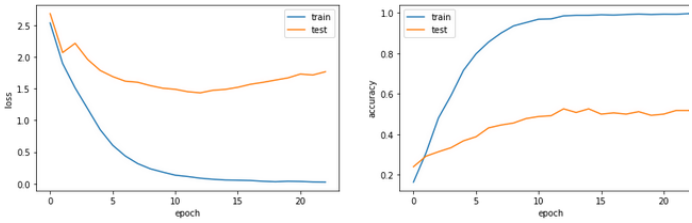


Figura 7: Gráficas de pérdida y accuracy del modelo 2

Finalmente, se planteó que el accuracy no aumentaba debido al tamaño de la imágenes que era muy pequeño para detectar algún patrón y no había una diferencia notable entre una característica entre una clase y otra. Debido a esto se decidió conseguir imágenes de mayor resolución por medio del scrapping de imágenes de Google. Debido a que el script del scrapping era cortado por parte de los servidores de Google debido a las múltiples llamadas, se consiguió un número disparejo de imágenes por cada clase. En la figura 8 se aprecia la cantidad por cada imagen.

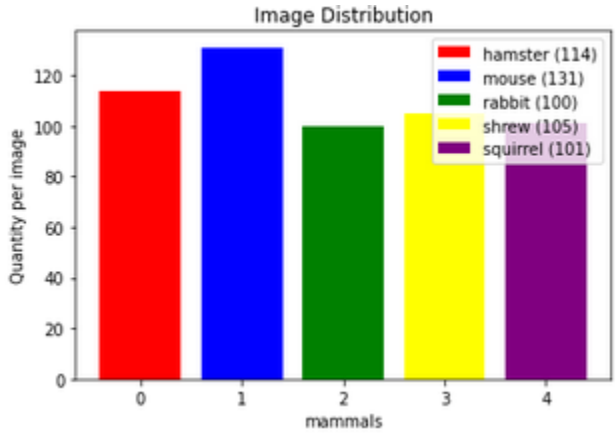


Figura 8: Cantidad de imágenes por cada clase para el modelo 3

Asimismo se verificó el tamaño de las imágenes y como se esperaba los tamaños eran muy variados, en la figura 9 se puede apreciar la distribución en el eje x e y. Por este motivo se realizó un resize a las imágenes para que sean de tamaño 224 x 224 píxeles.

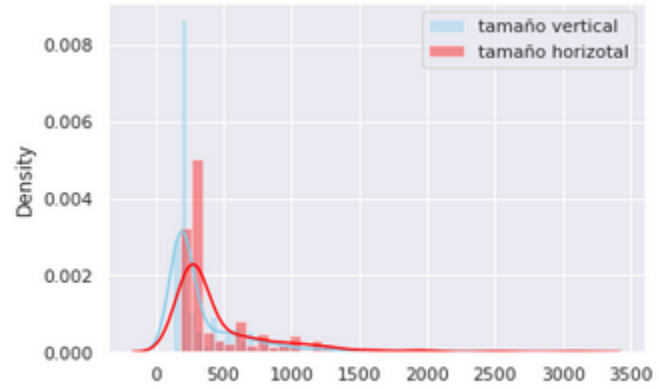


Figura 9: Distribución del tamaño de las imágenes horizontales y verticales para el modelo 3

Debido a la cantidad dispareja por cada clase se controló al momento de crear los dataset de entrenamiento y prueba, que los datos estuvieran balanceados por medio del parámetro stratify de la función train test split. De esta función se obtuvieron las siguiente distribución de datos

Train_dataset:	Test_dataset:
Hamster: 97	Hamster: 17
Ratón: 111	Ratón: 20
Conejo: 85	Conejo: 15
Musaraña: 89	Musaraña: 16
Ardilla: 86	Ardilla: 15

Luego se decidió implementar un modelo sin uso de transfer learning para verificar el desempeño del entrenamiento. En la figura 10, se aprecia el modelo propuesto, el cual cuenta con 88, 733 parámetros siendo el que menos tiene de los tres descritos. Los parámetros que se configuraron para el modelo fueron los siguientes:

initial_learning_rate = 0.001

epochs=100

batch_size=120

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0

activation (Activation)      (None, 224, 224, 3)       0

conv2d (Conv2D)              (None, 112, 112, 128)     3584

batch_normalization (BatchN  (None, 112, 112, 128)     512
ormalization)

activation_1 (Activation)    (None, 112, 112, 128)     0

dropout (Dropout)            (None, 112, 112, 128)     0

max_pooling2d (MaxPooling2D  (None, 56, 56, 128)       0
)

max_pooling2d_1 (MaxPooling  (None, 28, 28, 128)       0
2D)

conv2d_1 (Conv2D)            (None, 14, 14, 64)        73792

dropout_1 (Dropout)          (None, 14, 14, 64)        0

batch_normalization_1 (Batc  (None, 14, 14, 64)        256
hNormalization)

activation_2 (Activation)    (None, 14, 14, 64)        0

dropout_2 (Dropout)          (None, 14, 14, 64)        0

dense (Dense)                (None, 14, 14, 32)        2080

dropout_3 (Dropout)          (None, 14, 14, 32)        0

dense_1 (Dense)              (None, 14, 14, 16)        528

dropout_4 (Dropout)          (None, 14, 14, 16)        0

dense_2 (Dense)              (None, 14, 14, 8)         136

flatten (Flatten)            (None, 1568)              0

labels (Dense)               (None, 5)                 7845

=================================================================
Total params: 88,733
Trainable params: 88,349
Non-trainable params: 384
_________________________________________________________________
```

Figura 10: Resumen de las capas del modelo 3

El modelo llegó a tener un desempeño mayor llegando a un 75% de accuracy

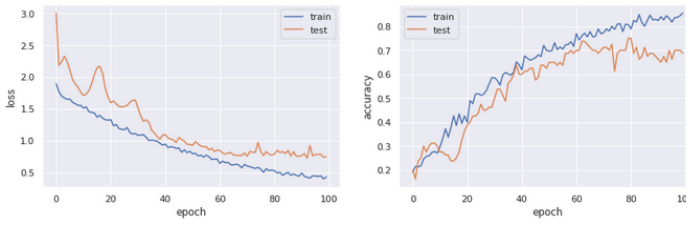


Figura 11: Gráficas de pérdida y accuracy del modelo 3

6. Discusión

6.1 Interpretación de los resultados

El implementar transfer learning, en vez de mejorar el desempeño de la red neuronal, el accuracy disminuyó, tal vez se deba a la complejidad de la red neuronal que no puede encontrar patrones en imágenes muy pequeñas.

Por otro lado, se puede apreciar que el accuracy mejoró al usar imágenes con mejor resolución llegando a casi alcanzar el 80%. Sin embargo la cantidad de imágenes era muy reducida comparado con los modelos que usaban el CIFAR-100, por lo que puede haber un sesgo.

6.2 Mejoras posibles del sistema

Se ha planteado que debido a la poca cantidad de imágenes en el último modelo, se generaría un sesgo. Por lo que usar técnica de data augmentation puede ayudar, pero seria mas recomendable aumentar el dataset con imágenes nuevas. También se puede llegar a tener datasets más equilibradas.

Si bien el CIFAR-100 tiene una cantidad aceptable de datos de prueba, tiene una limitación con respecto a subclases dentro de una clase. Ya que es más notable la diferencia entre un objeto de clase “frutas y vegetales” con respecto a “mamíferos pequeños”.