Entrega 2- Modelo de Clasificación de imágenes de Cámara Trampa

Julian David Santamaría Pabón
Universidad de Antioquia

Alejandro Ruiz
Universidad de Antioquia

1 Descripción de la Estructura del Notebook

El notebook entregado contiene varias secciones principales que incluyen la descarga y organización de los datos, la construcción y entrenamiento de un modelo de clasificación de imágenes, y la evaluación de resultados. A continuación, se presenta una descripción general de la estructura del notebook:

- **Descarga de Datos**: Se utilizan comandos de `wget` para descargar los datos desde una fuente en Zenodo. una plataforma reconocida para compartir datos científicos. Los datos descargados corresponden a imágenes recopiladas por el proyecto Serengeti, que utiliza cámaras trampa para capturar fotografías de la vida silvestre en su hábitat natural en el Parque Nacional del Serengeti [1, 2].

  Los conjuntos de datos incluyen tres archivos CSV que referencian las imágenes para los conjuntos de `entrenamiento` (train), `validación` (val) y `prueba` (test). Cada archivo CSV contiene metadatos importantes, como el identificador de la especie, que serán cruciales para el proceso de clasificación.

- **Organización de los Datos**: Una vez descargados, los datos deben organizarse de manera sistemática para facilitar su uso en el modelo de aprendizaje automático. En esta sección, se implementa un proceso automatizado para clasificar y distribuir las imágenes en carpetas específicas. Se crean directorios separados para los conjuntos de entrenamiento (train), validación (val) y prueba (test). Dentro de cada uno de estos directorios, las imágenes se agrupan en subcarpetas que corresponden al ID de especie al que pertenece cada imagen.

  Este enfoque de organización sigue las convenciones estándar utilizadas en frameworks de aprendizaje profundo, como TensorFlow y PyTorch, lo que permite una fácil integración y manipulación de los datos durante el proceso de entrenamiento y validación. Además, se implementan verificaciones para garantizar que no haya imágenes faltantes o duplicadas, y que cada imagen esté correctamente etiquetada según su clase.

- **Entrenamiento del Modelo**: Se define y entrena un modelo de clasificación de imágenes con una arquitectura específica (detallada en la siguiente sección).
- **Evaluación de Resultados**: Tras el entrenamiento, se evalúan la precisión, recall y F1 Score del modelo utilizando una matriz de confusión y otras métricas.

2 Descripción de la Solución

2.1 Arquitectura del Modelo

El modelo utilizado en este proyecto es una ResNet18 [3], una arquitectura ampliamente usada para tareas de clasificación de imágenes debido a su capacidad para aprender características profundas y complejas [4, 5]. Esta arquitectura consta de 18 capas, con bloques residuales que permiten la propagación eficiente de gradientes y facilitan el entrenamiento de redes profundas [3].

Para este proyecto, se emplearon los pesos preentrenados del modelo en el conjunto de datos ImageNet, utilizando la técnica de *transfer learning*. Esto permitió partir de una base de características ya aprendidas y adaptarlas al conjunto de datos específico de este proyecto, ahorrando tiempo de entrenamiento y mejorando la capacidad de generalización del modelo.

2.2 Preprocesado de Datos

El preprocesado de datos es una etapa clave para preparar las imágenes antes de ingresarlas al modelo. Para este proyecto, se aplicaron transformaciones específicas para los conjuntos de entrenamiento, validación y prueba:

- **Redimensionamiento**: Todas las imágenes se redimensionaron a 224x224 píxeles para cumplir con los requisitos de entrada de la ResNet18 [3].
- **Aumento de Datos (Data Augmentation)**: En el conjunto de entrenamiento, se aplicó una transformación de volteo horizontal aleatorio (`RandomHorizontalFlip`) para incrementar la diversidad del conjunto de datos y mejorar la robustez del modelo al enfrentar variaciones en la orientación de los animales en las imágenes.
- **Conversión a Tensores**: Las imágenes fueron convertidas a tensores mediante `ToTensor()`, un paso necesario para que puedan ser procesadas por PyTorch.
- **Normalización**: Cada imagen fue normalizada utilizando los valores de media [0.485, 0.456, 0.406] y desviación estándar [0.229, 0.224, 0.225], los cuales corresponden a los valores calculados en ImageNet. Esta normalización asegura que las características de las imágenes se encuentren en un rango adecuado para el modelo y ayuda a estabilizar el proceso de entrenamiento.

Las transformaciones fueron aplicadas de la siguiente manera:

- `train`: Redimensionamiento, volteo horizontal aleatorio, conversión a tensor y normalización.
- `val` y `test`: Redimensionamiento, conversión a tensor y normalización (sin aumento de datos).

3 Descripción de las Iteraciones

Debido a la alta demanda computacional, el tiempo total de entrenamiento del modelo fue de aproximadamente 3 horas, distribuidas en dos iteraciones (épocas). Dado el tiempo limitado, se trabajó con solo dos épocas, lo cual permitió observar un rendimiento inicial del modelo en los datos de entrenamiento y validación.

El entrenamiento y evaluación del modelo se realizó en un notebook de Google Colab utilizando una GPU T4 gratuita, lo que permitió acelerar significativamente el proceso de entrenamiento en comparación con el uso de CPU.

4 Descripción de los Resultados

4.1 Entrenamiento y Validación

El modelo alcanzó una precisión de entrenamiento de 85.76% y una precisión de validación de 91.49% en la primera época, mejorando en la segunda época a 91.61% en entrenamiento y 93.46% en validación. Estos resultados muestran una mejora continua en las métricas de rendimiento.

4.2 Evaluación en el Conjunto de Prueba

La precisión en el conjunto de prueba fue del 21.38%, con una matriz de confusión que refleja las predicciones del modelo en cada clase. Las métricas de precisión, recall y F1 Score fueron todas de 0.21, lo cual indica un desempeño bajo en la generalización del modelo.

Table 1: Informe de Clasificación del Conjunto de Prueba

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
0	0.875000	0.685315	0.768627	143
1	0.486842	0.552239	0.517483	67
2	0.787177	0.765368	0.776119	1,155
3	0.817045	0.922705	0.866667	2,691
4	0.769392	0.684914	0.724700	3,215
5	0.884615	0.953422	0.917731	13,268

...

Table 1

Clase	Precisión	Recall	F1-Score	Soporte
6	0.000000	0.000000	0.000000	4,297
7	0.001591	0.002844	0.002041	2,461
8	0.000000	0.000000	0.000000	209
9	0.000000	0.000000	0.000000	15,209
10	0.000075	0.000715	0.000136	1,398
11	0.000000	0.000000	0.000000	319
12	0.025862	0.003486	0.006143	2,582
13	0.000000	0.000000	0.000000	89
14	0.000809	0.002148	0.001175	931
15	0.071429	0.003846	0.007299	260
16	0.000000	0.000000	0.000000	467
17	0.000000	0.000000	0.000000	71
18	0.000000	0.000000	0.000000	1,287
19	0.000000	0.000000	0.000000	739
20	0.000000	0.000000	0.000000	133
21	0.000000	0.000000	0.000000	131
22	0.000000	0.000000	0.000000	486
23	0.000000	0.000000	0.000000	113
24	0.000000	0.000000	0.000000	759
25	0.000000	0.000000	0.000000	220
26	0.001361	0.004808	0.002121	208
27	0.000000	0.000000	0.000000	1,009
28	0.000000	0.000000	0.000000	78
29	0.000000	0.000000	0.000000	2,572
30	0.009709	0.000459	0.000877	4,355
31	0.000000	0.000000	0.000000	35
32	0.025202	0.001891	0.003517	13,224
33	0.000000	0.000000	0.000000	10,506
34	0.000000	0.000000	0.000000	34
35	0.000000	0.000000	0.000000	51
36	0.000000	0.000000	0.000000	709
37	0.000000	0.000000	0.000000	328
38	0.000000	0.000000	0.000000	0
39	0.000000	0.000000	0.000000	0
41	0.000000	0.000000	0.000000	0
42	0.000000	0.000000	0.000000	0
43	0.000000	0.000000	0.000000	0
45	0.000000	0.000000	0.000000	0
Exactitud	0.214453	0.214453	0.214453	0.214453
Promedio macro	0.108093	0.104185	0.104424	85,809
Promedio ponderado	0.209095	0.214453	0.209237	85,809

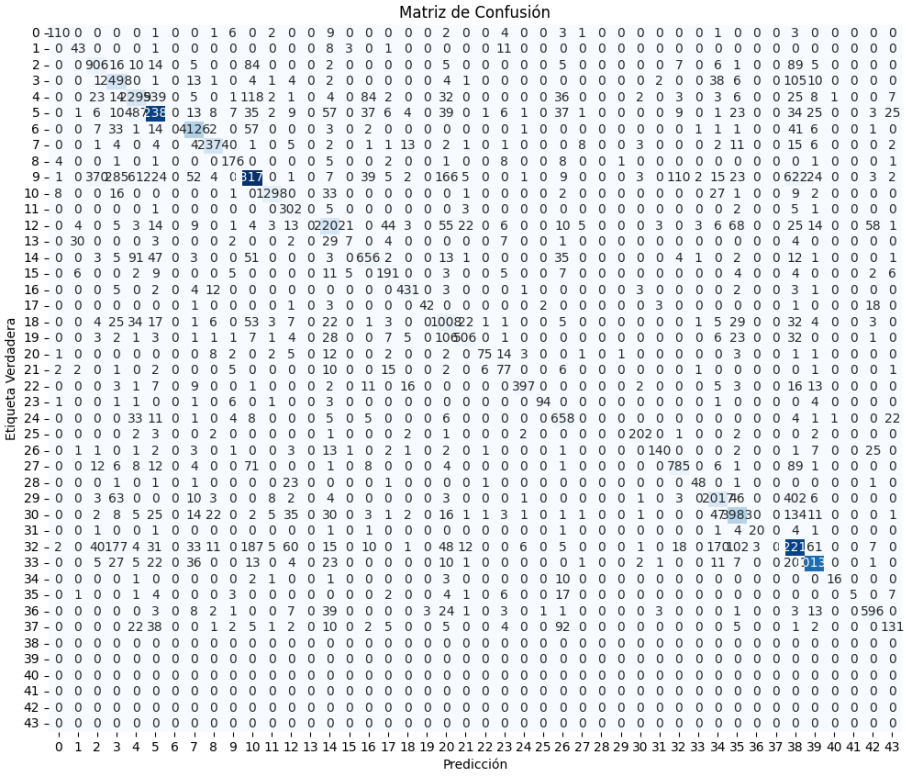

Figure 1: Matriz de Confusión del Conjunto de Prueba

5 Datos

Para obtener y hacer disponibles los datos en los procesos implementados en los notebooks, se deben seguir los siguientes pasos:

1. Descargar el archivo zip de la base de datos de Serengeti desde Zenodo y los archivos CSV correspondientes a los conjuntos `train`, `val` y `test`. Estos CSV contienen la información necesaria para clasificar cada imagen.

2. Descomprimir el archivo zip para obtener las imágenes, que estarán organizadas en una estructura de carpetas.

3. Leer cada archivo CSV para acceder a la columna que indica la ruta de

cada imagen dentro de la carpeta descomprimida y el ID de la especie correspondiente.

4. Crear una estructura de carpetas donde se generen las carpetas `train`, `val` y `test`. Dentro de cada una, se crean subcarpetas para cada especie utilizando el ID correspondiente.

5. Para cada entrada en los CSV, se copia la imagen en la subcarpeta correspondiente a su especie dentro de la carpeta designada para su conjunto (train, val o test).

6 Conclusión

- El modelo presentó buenos resultados en los conjuntos de entrenamiento y validación; sin embargo, en el conjunto de prueba mostró un bajo desempeño, evidenciado por la baja precisión y los valores de las métricas de evaluación.

- pesar de contar con un gran volumen de datos, el modelo no logró generalizar correctamente para el conjunto de prueba, lo que demuestra que la arquitectura utilizada no es lo suficientemente robusta para capturar patrones complejos en datos no vistos.

- La distribución desbalanceada de las clases impactó significativamente en el desempeño del modelo, ya que las clases con mayor cantidad de datos mostraron mejores valores de precisión y F1-Score. Esto refleja la necesidad de estrategias para abordar el desbalance de clases, como el uso de técnicas de sobremuestreo o ponderación de clases.

- Los resultados sugieren que es necesario realizar ajustes tanto en la arquitectura del modelo como en el preprocesamiento de los datos para mejorar su capacidad de generalización.

References

[1] V. Gabeff, M. Rußwurm, D. Tuia, and A. Mathis, "WildCLIP: Scene and animal attribute retrieval from camera trap data with domain-adapted vision-language models," *International Journal of Computer Vision*, 2024.

[2] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, "Data from: Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna," 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] A. Mahajan and S. Chaudhary, "Categorical image classification based on representational deep network (resnet)," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2019, pp. 327–330.

[5] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep learning in image classification using residual network (resnet) variants for detection of colorectal cancer," *Procedia Computer Science*, vol. 179, pp. 423–431, 2021.