1

Comparativa de Arquitecturas Livianas y Complejas para la Clasificación de Flores con CNNs

Rodríguez, Iván
a01781284@itesm.mx
Instituto Tecnológico Y De Estudios Superiores De Monterrey

***Abstracto*—Este artículo científico presenta la comparación de dos modelos de redes neuronales convolucionales (CNN) para la clasificación de cinco especies de flores: margarita, diente de león, rosa, girasol y tulipán. El primer modelo replica una arquitectura compleja propuesta por Prasad et al., mientras que el segundo es una versión optimizada y más ligera. Utilizando un conjunto de datos de aproximadamente 3,200 imágenes y técnicas de aumentación, se expandió el dataset a más de 9,500 imágenes. Tras el entrenamiento, el modelo simplificado alcanzó una precisión de validación del 79.25%, superando al modelo complejo en 9% y con 87% menos parámetros. Los resultados demuestran que, para tareas de clasificación con pocas clases, las arquitecturas livianas pueden ofrecer un mejor equilibrio entre rendimiento y eficiencia computacional.**

***Índice de términos*—Aprendizaje Automático, Aprendizaje Profundo, Clasificación de Flores, Clasificación de Imágenes, Convolutional Neural Networks, Deep Learning, Redes neuronales convolucionales.**

I. Introducción

En los últimos años, el reconocimiento de imágenes mediante técnicas de aprendizaje profundo se ha vuelto una herramienta crucial para resolver problemas de clasificación en distintos ámbitos, tales como, agricultura, medicina y seguridad. Una de las aplicaciones más interesantes y complejas es la clasificación de flores. Esto debido a la variabilidad que existe en color, forma, textura e iluminación de las imágenes.

Además de su valor visual, las flores tienen una importancia ecológica importante, ya que son los órganos reproductivos de la mayoría de las plantas y su rol es fundamental tanto en la reproducción como en la cadena alimenticia de cualquier ecosistema. Por estas razones, tener un buen conocimiento sobre estas mismas, es de suma importancia.

Las redes neuronales convolucionales (*CNN*) han demostrado ser altamente efectivas, dado que permiten la extracción de características importantes y distintivas de cada flor. Estas redes son capaces de detectar tanto patrones simples como complejos, lo cual las vuelve ideales para tareas de clasificación con múltiples flores (clases).

En este artículo se presenta la implementación de un modelo de una *CNN* inspirado en el artículo "*An efficient classification of flower images with convolutional neural networks*"[1], adaptado a la clasificación de 5 especies de flores: margarita (*daisy*), diente de león (*dandelion*), rosa (*rose*), girasol (*sunflower*) y tulipán (*tulip*). El objetivo principal es evaluar el desempeño de esta arquitectura con un conjunto de datos reducido y analizar si es necesaria una arquitectura tan compleja y robusta en un problema de 5 clases.

II. Trabajo Relacionado

La clasificación de flores mediante el uso de una *CNN* es un tema que ya ha sido explorado previamente en diversas ocasiones y utilizando distintas arquitecturas. A continuación, se presentan algunos de estos artículos:

1. Prasand et al. (2018)[1]
Propusieron una CNN utilizando 4 capas convolucionales con stochastic pooling, logrando un 97.78% de precisión con 132 clases de flores. El modelo utilizó capas convolutivas variables (16x16 a 5x5). Destacan que el aumento de datos (*data augmentation*) y el *stochastic pooling* mejoraron substancialmente los resultados, especialmente en imágenes con fondos complejos.

2. Mete & Ensari (2019)[2]
Combinaron *transfer learning* con algoritmos de *Machine Learning* clásicos (*Support Vector Machine, Random Forest*). Utilizaron aumento de datos y alcanzaron 99.8% de precisión (*Multilayer Perceptron*) y 98.5% (*Support Vector Machine*). Hacen énfasis en el uso de *CNN* preentrenadas para reducir la necesidad de arquitecturas personalizadas.

3. Patel & Patel (2020)[3]
Utilizaron *NAS-FPN* con *Faster R-CNN* para la clasificación. Además, hicieron uso del *transfer learning* y modelos preentrenados, logrando 96.2% mAP (*mean average precision*) en 30 clases y 87.6% mAP en 102 clases. Resaltan la necesidad de modelos complejos para imágenes con distintas flores y fondos con mucho ruido, a pesar de que estos consuman más recursos computacionales.

Los estudios utilizados como base de este trabajo coinciden en que:

- El aumento de datos es esencial para mejorar la precisión.
- Arquitecturas basadas en *transfer learning* y el uso de modelos preentrenados ofrecen un equilibrio entre complejidad y rendimiento.
- Los modelos complejos son perfectos para la

detección de subespecies, pero pueden ser excesivos para una clasificación básica.

III. Metodología

A. Conjunto de datos

Para poder empezar la construcción de la CNN correspondiente, primero se seleccionó un conjunto de datos (*dataset*) llamado "*Flowers Dataset*"[4], el cual se obtuvo de la página web *Kaggle,* este *dataset* consta de 2746 imágenes para entrenamiento (*train*) de 5 tipos de flores. Además de esto, igualmente contiene 924 imágenes para probar el modelo (*test*).

El *dataset* utilizado incluye las imágenes de entrenamiento divididas por clase en sus carpetas correspondientes. A pesar de esto, al hacer una revisión visual rápida de los datos se detectaron imágenes mal clasificadas o que no deberían aparecer en ese *dataset*. Para poder solucionar este problema se realizó una limpieza manual de los datos, revisando imagen por imagen y descartando imágenes que no fueran útiles para el entrenamiento.

TABLA I
Número de imágenes por clase, antes y después de la revisión

Clase	# de imágenes antes de la revisión manual	# de imágenes después de la revisión manual
Margarita	501	494
Diente de león	646	640
Rosa	497	414
Girasol	495	481
Tulipán	607	502
Total	2746	2531

Posterior a esto, se calculó el porcentaje de imágenes por clase que se obtuvieron, esto con la finalidad de verificar que el conjunto de datos seleccionados estuviera balanceado en cuanto a la cantidad de muestras por categoría. Mantener un balance en la distribución de clases es fundamental para evitar sesgos durante el entrenamiento del modelo.

TABLA II
Porcentaje de imágenes por distribución de clases

Clase	# de imágenes	Distribución
Margarita	494	19.52%
Diente de león	640	25.28%
Rosa	414	16.36%
Girasol	481	19.01%
Tulipán	502	19.83%
Total	2531	100%

Se realizó la separación del conjunto de datos de entrenamiento (*train*) y los datos con los que se validó (*validation*) el modelo. Se separaron aproximadamente el 20% de los datos totales de *train* (500 imágenes, 100 imágenes por cada clase de flor), y se movieron a una carpeta llamada *validation_data*.

Fig. 1 Muestra aleatoria de las imágenes del *dataset* utilizado.

Como se menciona anteriormente en la sección II. Trabajo Relacionado, varios artículos mencionan la importancia de realizar el aumento de datos (*data augmentation*) para aumentar la variabilidad del dataset y por ende obtener mejores resultados. Por esto mismo se aplicaron técnicas de *data augmentation* para generar dos imágenes distintas a partir de cada imagen original.

- Reescalada de los valores de píxeles al rango [0,1], dividendo cada valor entre 255.
- Rotación aleatoria de hasta 10 grados.
- Desplazamiento horizontal de hasta un 20% del ancho de la imagen.
- Zoom aleatorio de hasta un 30%.
- Volteo horizontal aleatorio.

B. Arquitecturas

Una vez preprocesado y organizado el conjunto de datos, se diseñaron e implementaron dos modelos de red neuronal convolucional (*CNN*). El primero se basa en la arquitectura propuesta por Prasad [1], la cual fue seleccionada por no requerir modelos preentrenados. El segundo es un modelo propio, diseñado con el objetivo de reducir la complejidad computacional manteniendo un rendimiento óptimo.

TABLA III
Arquitectura utilizada en el artículo científico [1]

Capa	Función	Parámetros Clave
Input	Input de la imagen	Redimensión a 128x128x3
Conv_1	Convolución	16x16; 32 filtros. Activación tanh
Conv_2	Convolución	9x9; 32 filtros. Activación tanh
Stoch_pooling_1	Pooling estocástico	2x2
Dropout_1	Dropout	No especificado
Conv_3	Convolución	5x5; 64 filtros. Activación tanh
Conv_4	Convolución	5x5; 64 filtros. Activación tanh
Stoch_pooling_2	Pooling estocástico	2x2
Dropout_2	Dropout	No especificado
Flatten_1	Aplanar	-
Dense_1	Capa densa	64 neuronas
Dropout_3	Dropout	No especificado

Dense_2 (Output)	Capa densa	Activación softmax

Existen diferentes técnicas de agrupación (*pooling*) que se utilizan normalmente en redes neuronales convolucionales, estas sirven para reducir el tamaño y mitigar el sobreajuste. Algunas de las más populares son el *max pooling* y el *pooling estocástico*. El *max pooling* consiste en seleccionar el valor máximo dentro de una región de la imagen, así destaca las características más fuertes. Por otro lado, el pooling estocástico (*stochastic pooling*) selecciona un valor aleatorio de la región de la imagen, esto ayuda a regularizar el modelo y evitar ligeramente el sobreajuste.

En este caso se optó por usar *max pooling* en lugar de pooling estocástico, ya que este último no se encuentra implementado de forma nativa en TensorFlow/Keras, y considerando el objetivo del trabajo y la búsqueda de simplicidad, *max pooling* resultó la alternativa perfecta.

Las capas convolucionales en las *CNN*s utilizan filtros (*kernel*) para obtener características específicas de las imágenes. El tamaño del filtro determina el tipo de características que puede detectar: filtros grandes (16x16 o 9x9) son efectivos para identificar patrones globales, mientras que filtros pequeños (3x3) se especializan en la detección de bordes y detalles más finos.

La arquitectura común de una CNN sigue un patrón donde las primeras capas utilizan filtros más grandes para capturar características generales, reduciendo gradualmente el tamaño de los filtros en capas más profundas para buscar detalles más específicos. Al mismo tiempo, el número de filtros se incrementa (16→32→64→128).

TABLA IV
ARQUITECTURA UTILIZADA PARA LA CREACIÓN DEL PRIMER MODELO

Capa	Función	Parámetros Clave
Input	Input de la imagen	Redimensión a 128x128x3
Conv_1	Convolución	16x16; 32 filtros. Activación tanh
Conv_2	Convolución	9x9; 32 filtros. Activación tanh
MaxPooling_1	Pooling máximo	2x2
Dropout_1	Dropout	25%
Conv_3	Convolución	5x5; 64 filtros Activación tanh
Conv_4	Convolución	5x5; 64 filtros Activación tanh
MaxPooling_2	Pooling máximo	2x2
Dropout_2	Dropout	25%
Flatten_1	Aplanar	-
Dense_1	Densa	64 neuronas. Activación tanh
Dropout_3	Dropout	50%
Dense_2 (Output)	Densa	5 neuronas. Activación softmax

A pesar de obtener resultados positivos en el artículo científico[1], la arquitectura utilizada es sumamente pesada, con un aproximado de 3,000,000 de parámetros.

Debido a que en este caso solo se clasifican 5 flores distintas, se diseñó un segundo modelo, mucho más liviano. A diferencia del primer modelo, en esta segunda arquitectura se decidió utilizar la función de activación *ReLU* en lugar de *tanh.* Esta decisión se tomó debido a que *ReLU* ofrece una mayor eficiencia computacional y mejor velocidad.

TABLA V
ARQUITECTURA UTILIZADA PARA LA CREACIÓN DEL SEGUNDO MODELO

Capa	Función	Parámetros Clave
Input	Input de la imagen	Redimensión a 128x128x3
Conv_1	Convolución	7x7; 16 filtros. Activación relu
MaxPooling_1	Pooling máximo	2x2
Conv_2	Convolución	5x5; 32 filtros. Activación relu
MaxPooling_2	Pooling máximo	2x2
Dropout_1	Dropout	25%
Conv_3	Convolución	3x3; 64 filtros Activación relu
Batch_norm_1	Normalización	-
MaxPooling_3	Pooling máximo	2x2
Conv_4	Convolución	3x3; 128 filtros. Activación relu
MaxPooling_3	Pooling máximo	2x2
Dropout_2	Dropout	25%
Flatten_1	Aplanar	-
Dense_1	Densa	64 neuronas. Activación relu
Dropout	Dropout	50%
Dense_2 (Output)	Densa	5 neuronas. Activación softmax

Para esta segunda arquitectura se consideraron las cuatro capas convolucionales que se utilizan en el artículo[1], además de la combinación de capas de *max pooling* y *dropout* La nueva arquitectura del segundo modelo resultó más liviana, utilizando alrededor de 400,000 parámetros.

IV. ENTRENAMIENTO Y PRIMEROS RESULTADOS

Teniendo listas las dos arquitecturas, primero se empezó con el entrenamiento del primer modelo (arquitectura basada en artículo[1]). Este modelo se entrenó con un *learning rate* de 0.001 durante 25 épocas. Los primeros resultados señalaron que tenía problemas de sobreajuste (*overfit*), dado que la precisión del modelo en el entrenamiento (*training accuracy*) fue de 86% y la precisión de validación fue de 68% (*validation accuracy*).

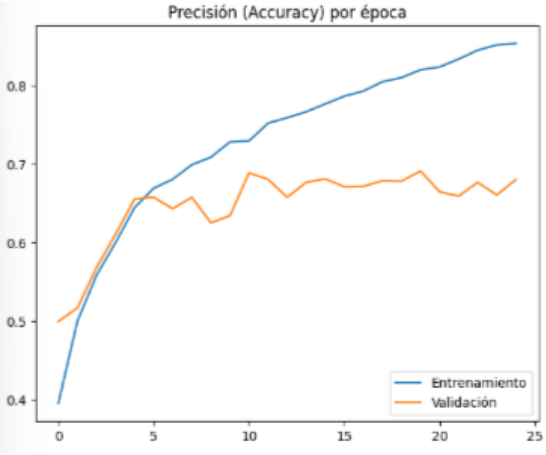

Fig. 2 Gráfico de la precisión de entrenamiento y validación por época del primer modelo.

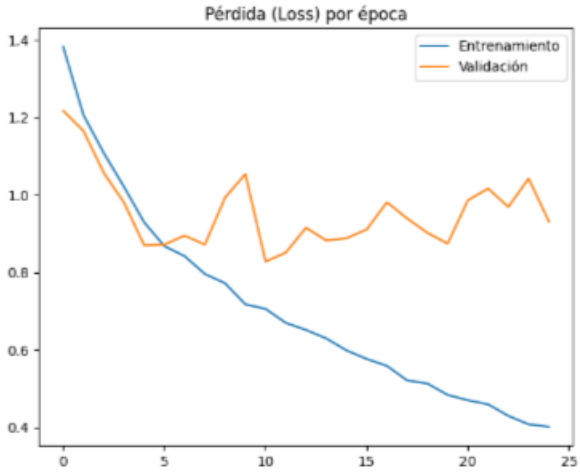

Fig. 3 Gráfico de pérdida de entrenamiento y validación por época del primer modelo.

Como se puede observar en las figuras 2 y 3, el modelo presentó buenos resultados con los datos de entrenamiento, pero a la hora de procesar los datos de validación hay una diferencia substancial en la precisión de entrenamiento y validación. Esto quiere decir que el modelo se está memorizando los datos de entrenamiento en lugar de aprender patrones para clasificar los tipos de flores.

Para intentar mejorar el rendimiento, se ajustó el *learning rate* a 0.0001 y para compensar el tener un *learning rate* más bajo, se aumentaron las épocas a 50. Sin embargo, después de reentrenar el modelo, los resultados no mejoraron, al contrario, el sobreajuste se intensificó.

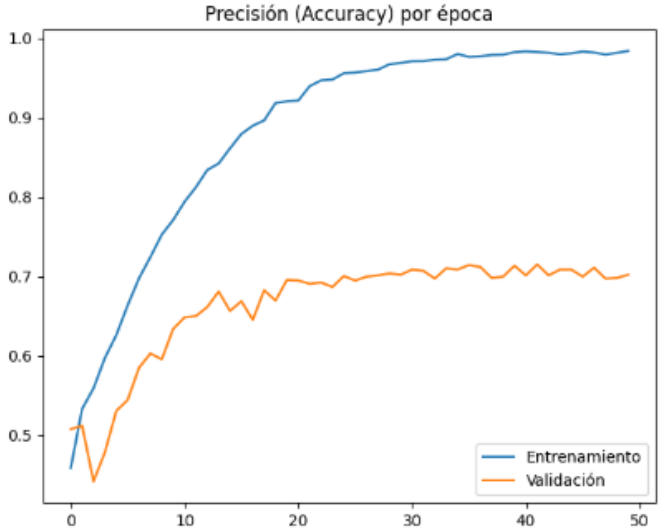

Fig. 4 Gráfico de precisión de entrenamiento y validación por época del primer modelo (ajuste de LR y épocas).

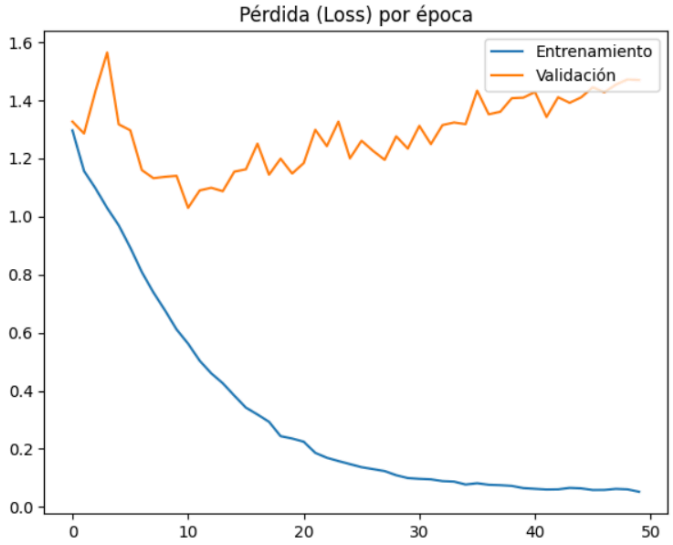

Fig. 5 Gráfico de pérdida de entrenamiento y validación por época del primer modelo (ajuste de LR y épocas).

Estos resultados indican que el problema no estaba solamente en los hiperparámetros (*learning rate* y épocas), sino en la capacidad del modelo frente a la diversidad del conjunto de datos.

Cabe recalcar que el modelo que estamos entrenando utiliza un *dataset* distinto y más pequeño, al que se menciona en el artículo[1]. Del mismo modo, considerando que clasificamos una menor cantidad de flores, se buscó complementar el *dataset* con diversas imágenes externas de otros sitios, esto con la finalidad de tener mayor variedad de imágenes de entrenamiento y mejorar la capacidad de generalización del modelo.

TABLA VI
CONJUNTO DE DATOS AJUSTADO

Clase	# de imágenes originales	# de imágenes aumentadas
Margarita	629	1258
Diente de león	640	1280
Rosa	737	1474
Girasol	635	1270
Tulipán	631	1262
Total	3272	6544

Es importante mencionar que, aunque ambos modelos fueron entrenados nuevamente con el conjunto de datos ajustado, el impacto fue distinto. Al reentrenar el modelo basado en el artículo [1], no se observó una mejora significativa, e inclusive el rendimiento empeoró. Por el contrario, con el segundo modelo (más ligero), mostró una mejora notable en sus métricas. Aunque no se cuenta con las gráficas previas del segundo modelo entrenado con el dataset reducido, se observó claramente una mayor estabilidad durante el entrenamiento y una mejor precisión final.

Para entrenar la segunda arquitectura se utilizó un *learning rate* de 0.0001, con la intención de poder evitar otro posible caso de *overfit*. Adicionalmente, se utilizaron 35 épocas.

Al finalizar el entrenamiento, se obtuvo una precisión de entrenamiento de 86.66% y una precisión de validación del 79.25%.

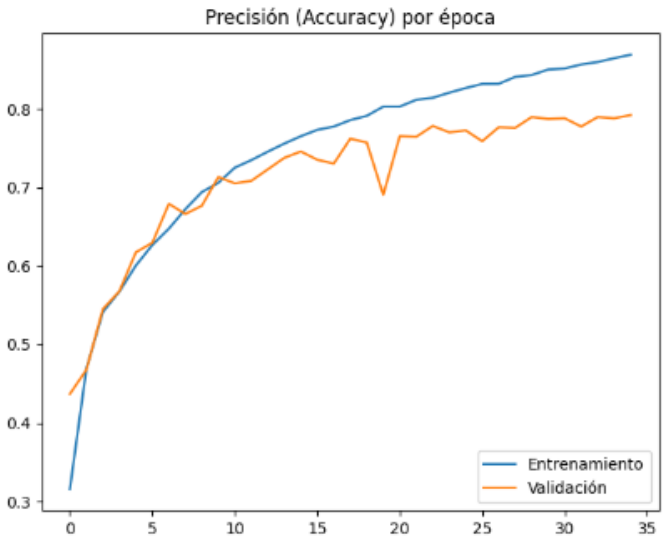

Fig. 6 Gráfico de precisión de entrenamiento y validación por época del segundo modelo.

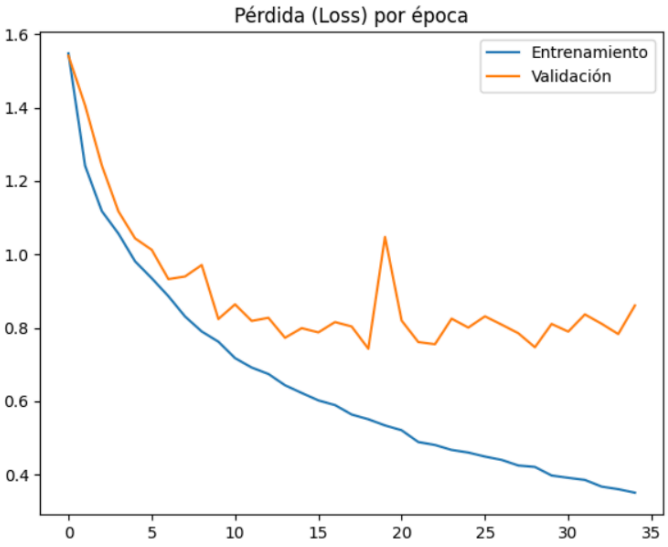

Fig. 7 Gráfico de pérdida de entrenamiento y validación por época del segundo modelo.

Como se puede observar en las figuras 6 y 7, se obtuvieron resultados notablemente mejores a comparación del primer modelo entrenado. Se observa que la perdida de validación es mayor a la perdida de entrenamiento, especialmente en las últimas épocas. Esto, junto con la diferencia de aproximadamente 7.4% entre la precisión del entrenamiento (86.6%) y la de validación (79.2%), sugiere la existencia de un pequeño sobreajuste (*overfit*).

V. Resultados y discusión

Posterior a la generación de gráficas donde se compara precisión de entrenamiento con precisión de validación y la pérdida de entrenamiento con la pérdida de validación, se procedió a evaluar el rendimiento del modelo mediante una matriz de confusión utilizando los datos de prueba (*test*).

Para la evaluación del modelo se utilizaron métricas estándar de clasificación. La precisión mide la proporción de predicciones correctas para cada clase. Responde a la pregunta: "De todas las imágenes que el modelo clasificó como pertenecientes a un tipo de flor, ¿cuántas realmente correspondían a este tipo?"[5]. La precisión es calculada por la siguiente ecuación:

$$\text{Precisión} = \frac{TP}{TP + FP}$$

Ecuación 1 Fórmula para calcular la precisión.

Donde TP (*True Positives*) representa instancias correctamente clasificadas de la clase y FP (*False Positives*) representa instancias incorrectamente clasificadas de la clase.

El *recall* indica la capacidad del modelo para identificar correctamente las instancias positivas reales. Responde a la pregunta: "De todas las imágenes que realmente pertenecen a una especie particular, ¿cuántas fueron correctamente

identificadas por el modelo?"[5]. El *recall* es calculado por la siguiente ecuación:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Ecuación 2 Fórmula para calcular el *recall*.

Donde TP representa instancias correctamente clasificadas de la clase y FN representa instancias de la clase que fueron incorrectamente clasificadas como otra clase.

El *f1-score* representa la media armónica entre precisión y *recall*, equilibrando ambas métricas en un solo valor. Esta métrica es calculada por la siguiente ecuación:

$$\text{F1-Score} = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}}$$

Ecuación 3 Fórmula para calcular el *F1-Score*.

A continuación se presenta la matriz de confusión para el modelo basado en la arquitectura de Prasand et al., después de haber hecho los cambios al conjunto de datos para enriquecerlo más.

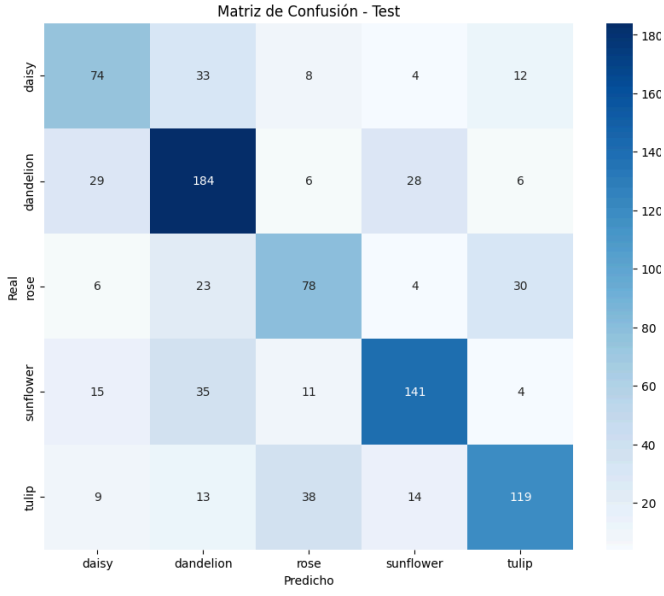

Fig. 8 Matriz de confusión del primer modelo.

La figura 8 nos muestra el rendimiento entre clases que tuvo nuestro primer modelo. La clase *dandelion* mostró el mejor desempeño con 184 predicciones correctas y las mejores métricas en general (precisión: 0.64, recall: 0.73, f1-score: 0.68). *Sunflower* también demostró un buen rendimiento con 141 clasificaciones correctas. *Tulip* logró tener 119 predicciones correctas, pero demostró confundirse con *rose* (38 casos). Por su parte, *daisy* presentó confusiones con múltiples clases, especialmente *dandelion*. La clase rose tuvo las mayores dificultades, solo obtuvo 78 predicciones correctas y tuvo las métricas más bajas (precisión: 0.55, recall: 0.55, f1-score: 0.55).

TABLA VII
MÉTRICAS DEL PRIMER MODELO

Clase	Precisión	Recall	F1-score
Margarita	0.56	0.56	0.56
Diente de león	0.64	0.73	0.68
Rosa	0.55	0.55	0.55
Girasol	0.74	0.68	0.71
Tulipán	0.70	0.62	0.65

A pesar de que el modelo 1 mostró resultados sólidos al inicio, empeoró después de expandir el conjunto de datos. Esto podría ser debido a la complejidad arquitectónica. Con más de 3 millones de parámetros, el modelo es capaz de memorizar los datos de entrenamiento. Al introducir una mayor cantidad de datos al modelo, no logró generalizarlos adecuadamente y presentó un mayor sobreajuste. Además, la falta de técnicas de regularización, como *batch normalization*, pudieron haber influido en la pérdida de validación. Con estos resultados podemos observar que, sin un ajuste correcto, las arquitecturas complejas pueden ser contraproducentes cuando se tiene un conjunto de datos limitado.

A continuación se presenta la segunda matriz de confusión, generada a partir del segundo modelo con el conjunto de datos de prueba (*test*).

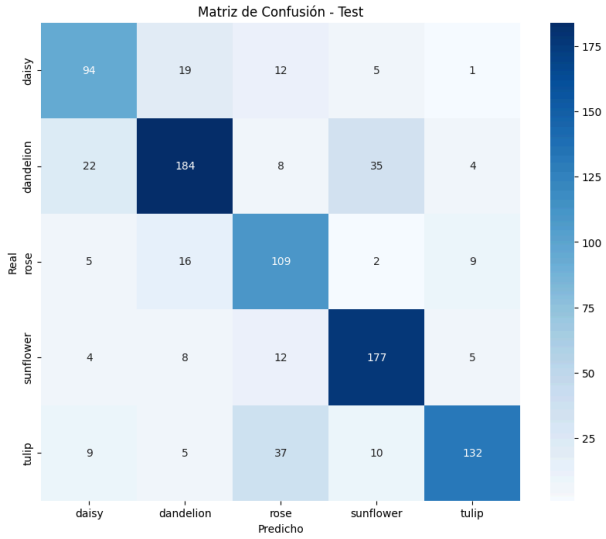

Fig. 9 Matriz de confusión del segundo modelo.

El modelo demostró un rendimiento considerablemente superior al anterior, con predicciones correctas más altas en todas las clases: *daisy* (94), *dandelion* (184), *rose* (109), *sunflower* (177) y *tulip* (132). Las clases *dandelion* y *sunflower* se siguieron desempeñando de buena manera, mientras que *tulip* obtuvo una mejora notable a comparación del modelo anterior. Igualmente, la clase *daisy* logró una reducción significativa en la confusión con *dandelion*. Las métricas del segundo modelo confirman el rendimiento superior que es observable en la matriz de confusión. *Tulip* tuvo la mayor precisión (0.87), seguido de *dandelion* (0.79) y *sunflower* (0.77). *Sunflower* demostró la mejor capacidad para

identificar todas sus instancias reales (*recall*). Los *f1-scores* nos indican un rendimiento equilibrado en las clases *sunflower* (0.81) y *tulip* (0.77). La clase rose presentó las peores métricas (precisión: 0.61, *recall*: 0.77, *f1-score*: 0.68), esto nos indica que tiene dificultades para diferenciar esta flor.

TABLA VIII
MÉTRICAS DEL SEGUNDO MODELO

Clase	Precisión	Recall	F1-score
Margarita	0.70	0.72	0.71
Diente de león	0.79	0.73	0.76
Rosa	0.61	0.77	0.68
Girasol	0.77	0.86	0.81
Tulipán	0.87	0.68	0.77

VI. CONCLUSIÓN

Este trabajó comparó dos arquitecturas de redes neuronales convolucionales para la clasificación de cinco especies de flores, demostrando que no siempre la mayor complejidad se traduce en un mejor rendimiento, especialmente en problemas de clasificación con pocas clases.

Comparación de modelos:

Modelo 1 (Basado en Prasad et al.[1]):

- Arquitectura compleja con 3,000,000 de parámetros.
- Precisión de validación: 70% (después de utilizar el dataset ampliado).
- Gran problema de sobreajuste (diferencia de 28% entre entrenamiento y validación).
- Uso de funciones de activación *tanh* y pooling estocástico.

Modelo 2 (Arquitectura simplificada):

- Arquitectura optimizada con 400,000 parámetros (87% menos parámetros)
- Precisión de validación: 79.25%
- Mucho menor sobreajuste (diferencia de 7.4% entre entrenamiento y validación).
- Uso de funciones de activación *ReLU* y max pooling para mayor eficiencia computacional.

Uno de los hallazgos principales y más evidente fue la eficiencia de arquitecturas simples. El segundo modelo (simple) superó al complejo en precisión de validación por un 9%. Esto demuestra que para problemas de cinco clases, las arquitecturas menos complejas pueden ser más efectivas. La expansión del dataset mejoró significativamente el rendimiento del modelo simple, mientras que el modelo complejo no se benefició del mismo modo.

Con este trabajo se demostró que la hipótesis de "mayor complejidad = mejor rendimiento" no siempre es válida en deep learning. Para la clasificación de flores con pocas clases, una arquitectura bien diseñada, pero simple, puede superar significativamente a modelos más complejos, obteniendo una mejor precisión, menor sobreajuste y mayor eficiencia computacional.

VII. REFERENCIAS

[1] M. Prasad *et al.*, "An efficient classification of flower images with convolutional neural networks," *International Journal of Engineering & Technology*, vol. 7, no. 1, pp. 384–391, 2018, Accessed: Jun. 01, 2025. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/80918318/3444-libre.pdf?1645009408=&response-content-disposition=inline%3B+filename%3DAn_efficient_classification_of_flower_im.pdf&Expires=1748813341&Signature=Z51Z2d6WTmBmjlihUvC6Nz2aRSy03XjV~-ig2gjs~ZqWRXFJMUfats8sTdt7TwI-tlQKA4~aqHoGszejg09nWrXB88~pK-b94Thq6whFpYhScksD2aCE1m-lORqdvCj5l8YW9V9Kur3TFhXcLQ44vkdiXxP5e7DadxHfonv6jV-hQEO~8r-jgf8AB5dTmawSOSFtFus~4-~z7vivnoxfhQP0u1-qN1W9BfnuaOJRJqvQDcIF2-Z70G9RMPNRpbgBmxrVwCuYN9k9mmkA20NO1kCN-NYNnpVI5SLS7uS8IufyhwH7Vh10PG~oHG6vSRxc7ikKu1zx3hAYKA-DdacLkg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[2] B. R. Mete and Tolga Ensari, "Flower Classification with Deep CNN and Machine Learning Algorithms," *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2019, doi: https://doi.org/10.1109/ismsit.2019.8932908.

[3] I. Patel and S. Patel, "An optimized deep learning model for flower classification using NAS-FPN and faster RCNN," *International Journal of Scientific & Technology Research*, vol. 9, no. 3, pp. 5308–5318, Mar. 2020, Accessed: Jun. 01, 2025. [Online]. Available: https://www.researchgate.net/publication/344248895_An_optimized_deep_learning_model_for_flower_classification_using_NAS-FPN_and_faster_RCNN

[4] S. Gupta, "Flowers Dataset," *Kaggle.com*, 2021. https://www.kaggle.com/datasets/imsparsh/flowers-dataset (accessed Jun. 02, 2025).

[5] P. Kashyap, "Understanding Precision, Recall, and F1 Score Metrics," *Medium*, Dec. 02, 2024. https://medium.com/@piyushkashyap045/understanding-precision-recall-and-f1-score-metrics-ea219b908093