Santiago Jaramillo Ocampo
Andrés Felipe Castro Monsalve
Fundamentos de Deep Learning

Rice Classification CNN

En el presente documento se dará una breve explicación acerca del proceso que se siguió para el desarrollo de este proyecto. Además de una demostración de los mejores resultados obtenidos.

Lo primero que se realizó fue una exploración del dataset que íbamos a utilizar, donde se analizaron cada una de las clases de arroz que se habían y la cantidad de imágenes que hay por cada uno de las clases, para de esta manera poder tomar la mejor decisión a la hora de distribuir los datos en entrenamiento, test y validación. Además dimos un vistazo del tipo de imágenes que hay para cada uno de los tipos de arroz.

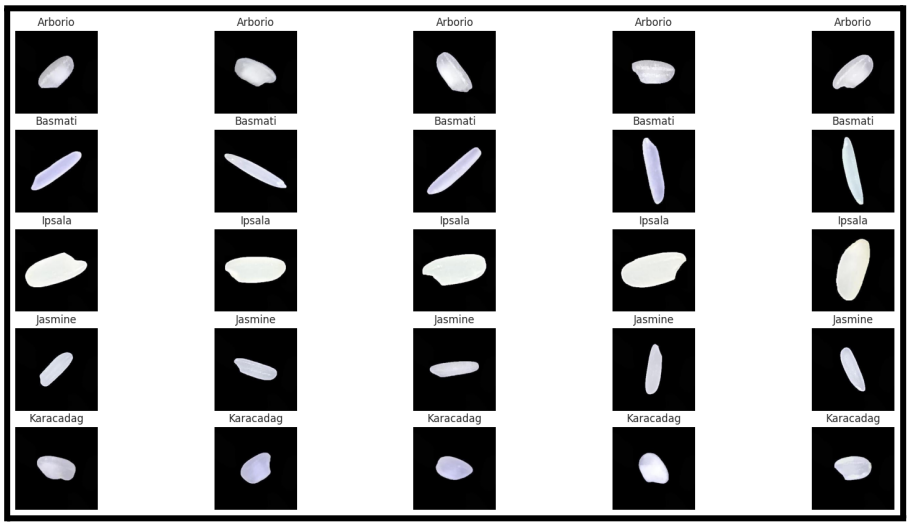


Imagen 1. Vistazo de imágenes que hay en el dataset.

Ya conociendo los datos que íbamos a utilizar en el proceso de entrenamiento empezamos a desarrollar cada uno de los pipelines donde se iban a desarrollar diferentes modelos de redes neuronales convolucionales y diferentes arquitecturas para determinar cuál de ellos es mejor.

La estructura usada para cada uno de los pipelines fue la siguiente:

- Importación de las librerías necesarias para el desarrollo del proyecto.
- Carga de los datos, en nuestro caso los datos son descomprimidos de un zip que se le pasa al proyecto.
- Separación de los datos en los diferentes en los grupos de entrenamiento, test y validación. Se usó un 80% de los datos para el entrenamiento de las redes neuronales, 10% para el proceso de test y 10% para el proceso de validación del modelo ya entrenado.
- Entrenamiento del modelo donde se definieron diferentes arquitecturas y se obtuvieron gráficas sobre el proceso el comportamiento del accuracy y la pérdida o fallos que tenía el proyecto durante cada una de las épocas definidas. Además se implementó una matriz de confusión para ver el rendimiento del modelo en el proceso de entrenamiento.
- Por último se implementó el proceso de validación del modelo con datos que nunca había visto para poder ratificar su rendimiento.

En la configuración de los modelos, se siguió una estructura similar en casi todos, comenzando con la normalización de las imágenes para que cada uno de los píxeles tomara valores entre 0 y 1.

Se utilizaron capas de convolución 2D junto con MaxPool2D para extraer las características de cada imagen y reducir la dimensión de la salida de una capa a otra. Se empleó la función de activación ReLU, se utilizó la operación de flatten para aplanar las imágenes y tenerlas en un solo vector. Además, en algunos de los modelos se incorporó la técnica de dropout para que ciertas neuronas "ignoraran" valores que no contribuyen significativamente a una detección eficiente de imágenes, evitando así el sobreajuste. Se implementaron varias capas densas en diferentes partes del modelo.

En el modelo que se presenta a continuación, se empleó una capa de convolución 2D con su correspondiente MaxPooling, utilizando la función de activación ReLU. Cada capa densa consta de 100 unidades, se utilizaron 6 filtros en la capa de convolución 2D, y se entrenó el modelo durante 6 épocas.

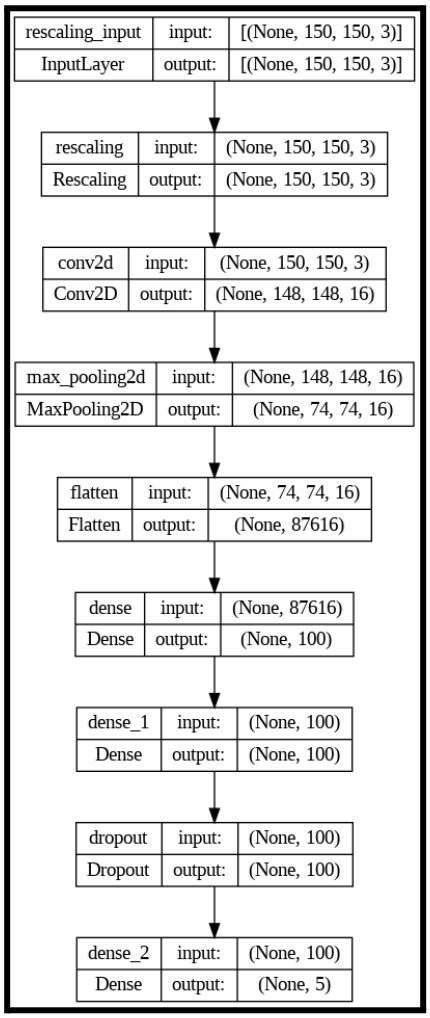


Imagen 2. Arquitectura del modelo.

Datos

Para disponibilizar los datos de la solución en cualquier notebook en Google Colab, realizar lo siguiente:

1. Descargar dataset comprimido (en zip) desde: https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset
2. Subir el archivo (con nombre “archive.zip”) a los archivos del ambiente de ejecución del notebook en Google Colab. Fijarse que se haya completado la subida antes de iniciar la ejecución de cualquier notebook.

De esta manera queda disponible el dataset para la ejecución de cualquiera de los notebooks de la solución, los cuales se encuentran en el repositorio.

Iteraciones y resultados

Modelo 1

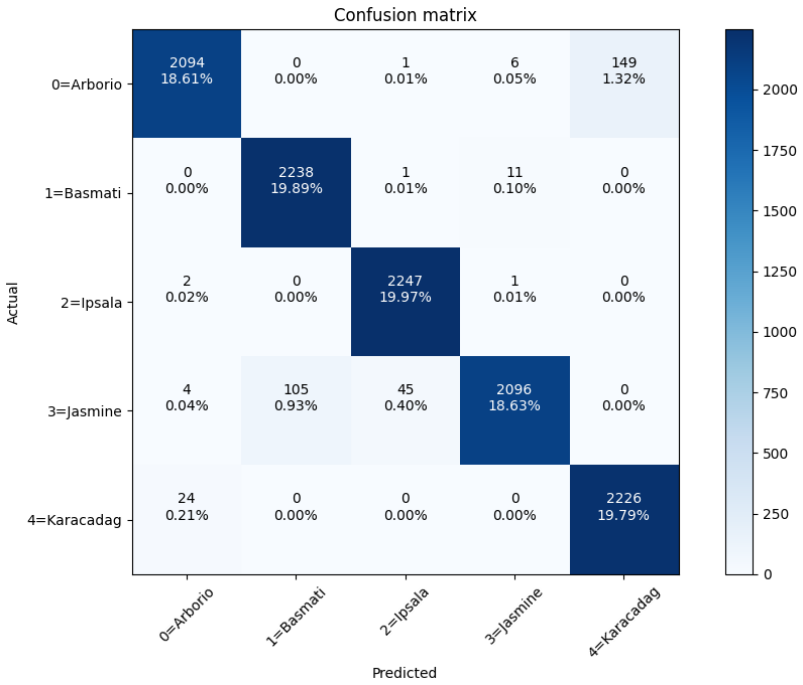
El primer modelo se diseñó con la intención de crear rápidamente una versión funcional y sencilla para abordar el problema de detección de tipos de arroz. Se adoptó una estrategia inicial de implementar parámetros modestos en las capas principales con el objetivo de tener un punto de partida funcional y luego iterar sobre él para realizar mejoras progresivas.

Arquitectura del Primer Modelo:

- Capa de Entrada (Input): Rescaling y Normalización de las imágenes. Capa Convolucional: Se emplearon solamente 2 filtros con un tamaño de kernel de 2x2 y una función de activación ReLU. Esta elección inicial se hizo con la idea de mantener la complejidad baja.
- Capa de Pooling: MaxPooling con un pool size de 10x10 y strides de 10. Se optó por un tamaño de pool grande inicialmente para simplificar la estructura y obtener una representación más general de las características.
- Capa de Aplanado (Flatten): Para transformar los datos en un formato adecuado para la capa densa.
- Capa Densa (Fully Connected): Con 20 unidades y una función de activación ReLU. Esta capa se mantuvo relativamente pequeña en términos de unidades para limitar la complejidad inicial.

Resultados:

Clase	**Precisión**	**Recall**	**F1-score**
Arborio	0.99	0.93	0.96
Basmati	0.96	0.99	0.97
Ipsala	0.98	1.00	0.99
Jasmine	0.99	0.93	0.96
Karacadag	0.94	0.99	0.96
Promedio	0.97	0.97	0.97



El modelo destaca en la precisión de Arborio y Jasmine, pero necesita mejorar el recall. Basmati muestra buen equilibrio, con oportunidad de ajustar para un mejor F1-Score. Ipsala tiene un excelente rendimiento. En el caso de Karadag, la precisión es la más baja, siendo propenso a confusiones con Arborio.

Áreas de Mejora:

- Aumento del Número de Filtros y Capas Convolucionales: Para capturar patrones más complejos en las imágenes.
- Reducción del Tamaño del Pooling y Strides: Para preservar más detalles durante la fase de pooling.
- Ajuste del Número de Unidades en la Capa Densa: Explorar tamaños más grandes para permitir al modelo aprender representaciones más complejas.

Modelo 2

El segundo modelo representa una evolución del primer modelo, con ajustes destinados a mejorar su rendimiento. Aunque se lograron mejoras en las métricas de clasificación, se observa cierto nivel de sobreajuste.

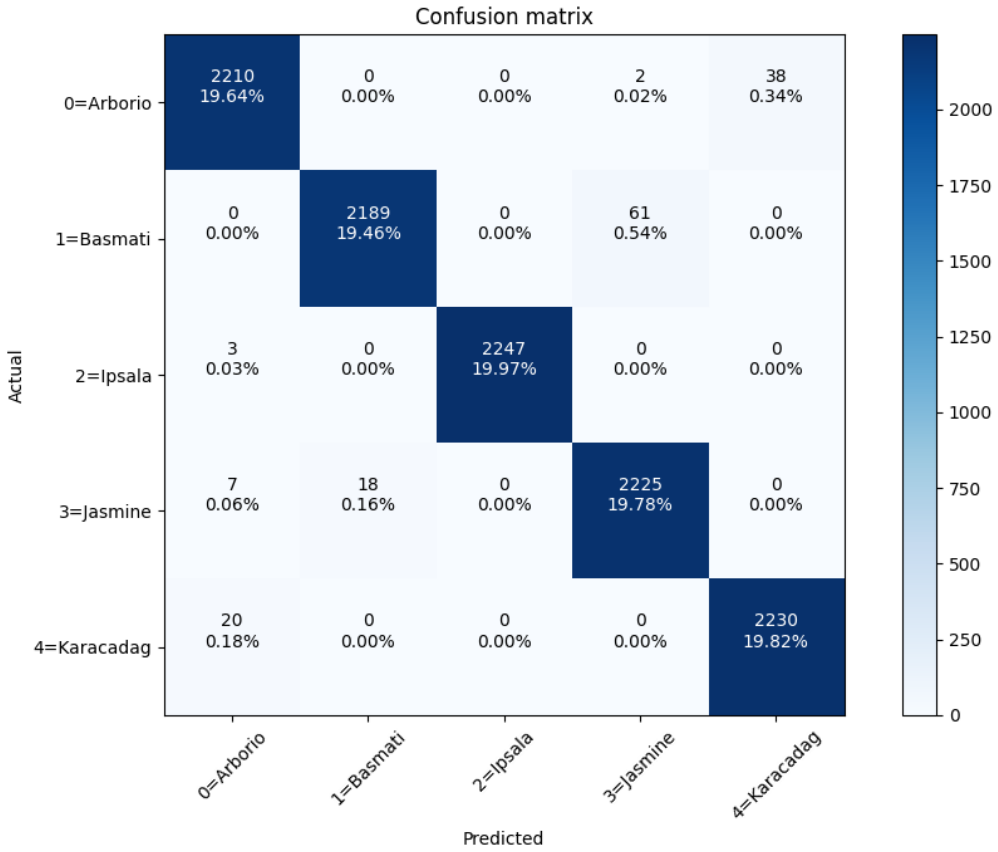
Arquitectura del Segundo Modelo:

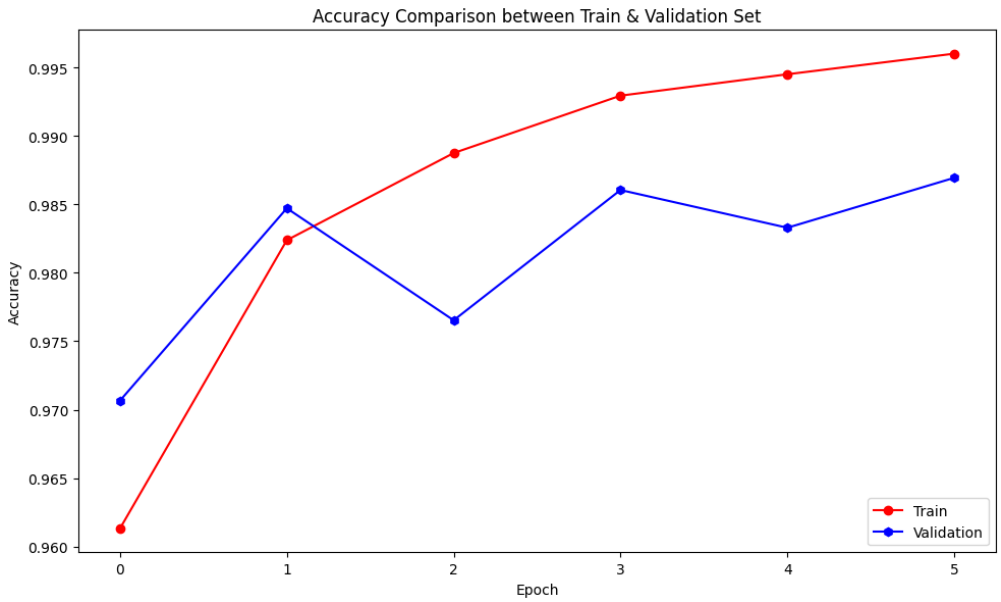
- Capa de Entrada (Input): Rescaling y Normalización de las imágenes.
- Capa Convolucional: Se aumentó el número de filtros a 16, se utilizó un tamaño de kernel de 3x3 y se mantuvo la función de activación ReLU. Esto busca capturar patrones más complejos en las imágenes.
- Capa de Pooling: Se redujo el tamaño del pool a 2x2 y los strides a 2 para preservar más detalles durante la fase de pooling.
- Capa de Aplanado (Flatten): Para transformar los datos en un formato adecuado para la capa densa.
- Capa Densa (Fully Connected): Se incrementó el número de unidades a 100 y se mantuvo la función de activación ReLU.
- Capa de Salida: 5 unidades con función de activación lineal, ya que se trata de un problema de clasificación multiclase.

Resultados:

Test accuracy: 0.9868

Clase	Precisión	Recall	F1-score
Arborio	0.99	0.98	0.98
Basmati	0.99	0.98	0.98
Ipsala	1.00	1.00	1.00
Jasmine	0.97	0.99	0.98
Karacadag	0.98	0.99	0.99
Promedio	0.99	0.99	0.99





El Modelo 2 representa una mejora con respecto al Modelo 1, logrando resultados superiores en las métricas de clasificación. Sin embargo, se evidencia un nivel de sobreajuste.

Áreas de Mejora:

- Regularización: Considerar la incorporación de técnicas de regularización para abordar el sobreajuste observado.

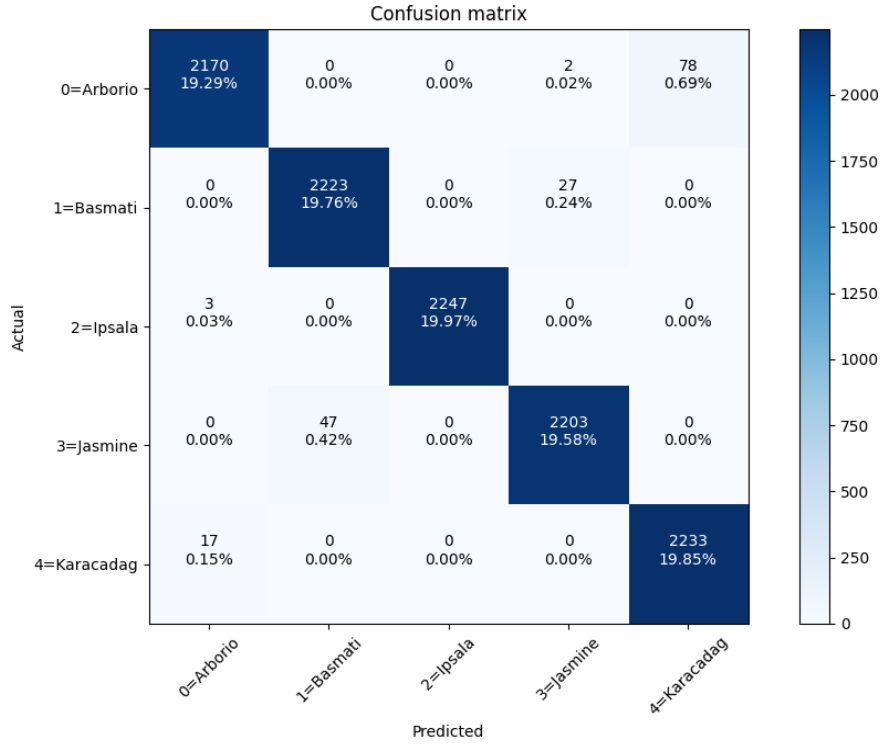
Modelo 3

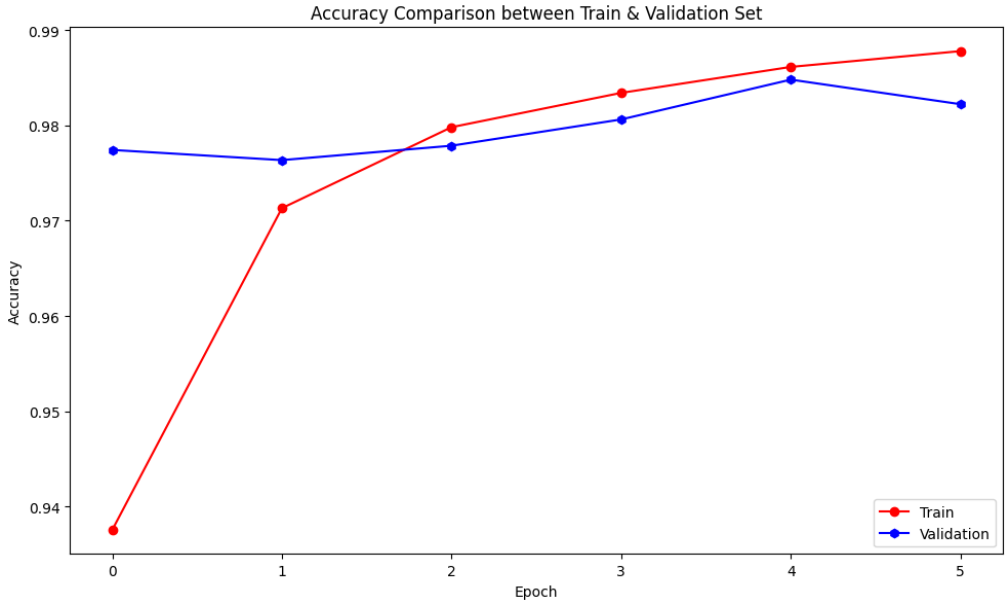
El tercer modelo busca generar una mejora respecto al Modelo 2, enfocándose en la reducción del sobreajuste. La única diferencia respecto al segundo modelo es la introducción de una capa Dropout, con una tasa del 0.5, para reducir el sobreajuste al desactivar aleatoriamente algunas neuronas durante el entrenamiento.

Resultados:

Test accuracy: 0.9845

Clase	**Precisión**	**Recall**	**F1-score**
Arborio	0.99	0.96	0.97
Basmati	0.97	0.99	0.98
Ipsala	1.00	1.00	1.00
Jasmine	0.98	0.97	0.98
Karacadag	0.96	1.00	0.98
Promedio	0.98	0.98	0.98





El tercer modelo demuestra una mejora significativa en la reducción del sobreajuste, sin embargo, existe una mínima reducción en el performance de las métricas de clasificación y del accuracy evaluada en los datos de test (precisamente de 0.0023). Dada la pequeña diferencia en la precisión del conjunto de prueba y la mejora en la gestión del sobreajuste en el Modelo 3, este modelo podría ser preferido por su capacidad de generalización ante datos desconocidos.

Áreas de Mejora:

- Optimización de Hiperparámetros: Ajustar otros hiperparámetros para observar si se puede mejorar aún más el rendimiento y la capacidad de generalización.

Modelo 4

El modelo 4 tiene casi la misma arquitectura del modelo anterior, solo que se agregó la función kernel_regularizer en la capa densa con el fin de regularizar los pesos de las capas con el fin prevenir el sobreajuste.

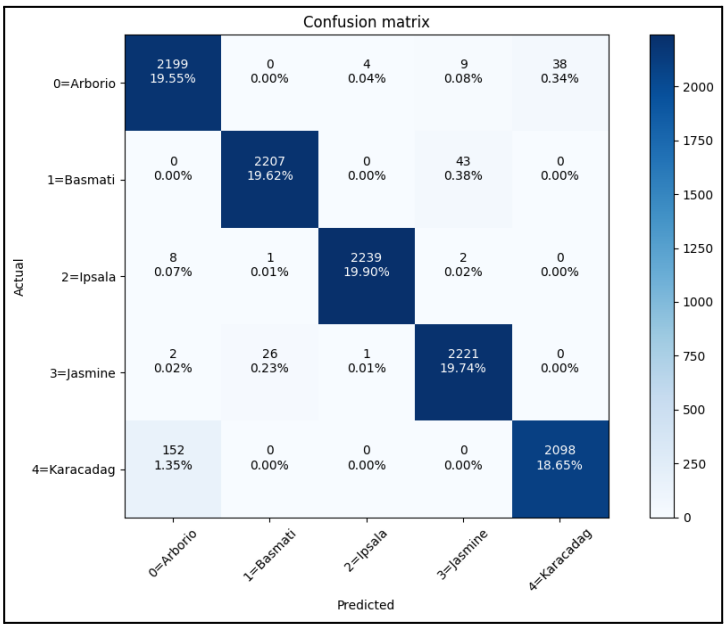
Arquitectura del cuarto Modelo:

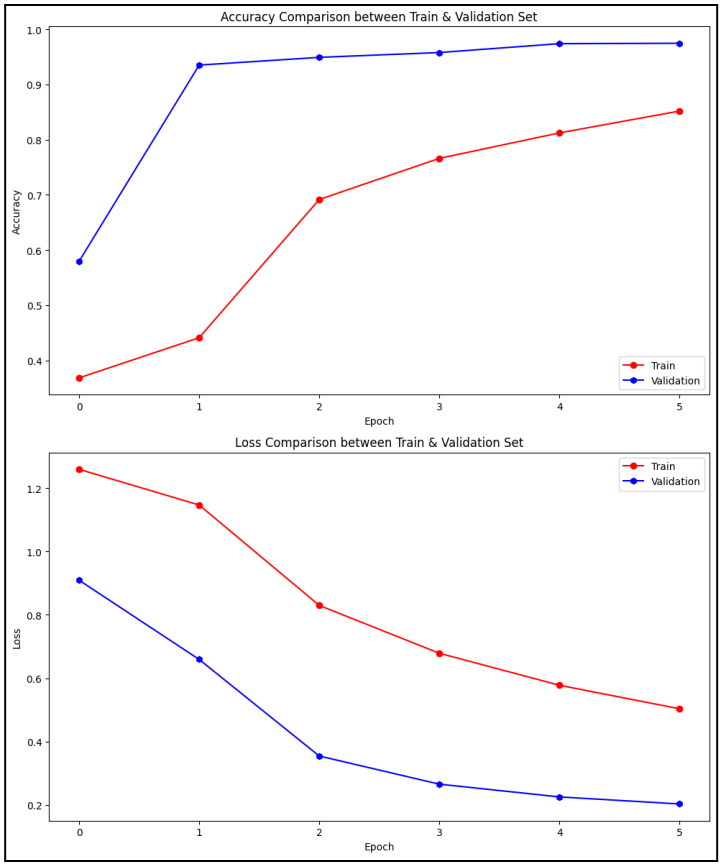
- Capa de Entrada (Input): Rescaling y Normalización de las imágenes.
- Capa Convolucional: Se aumentó el número de filtros a 16, se utilizó un tamaño de kernel de 3x3, se mantuvo la función de activación ReLU y se le agregó la función kernel_regularizer con un valor de 0.001.
- Capa de Pooling: Se redujo el tamaño del pool a 2x2 y los strides a 2 para preservar más detalles durante la fase de pooling.
- Capa de Aplanado (Flatten): Para transformar los datos en un formato adecuado para la capa densa.
- Capa Densa (Fully Connected): Se incrementó el número de unidades a 100 y se usó la función de activación softmax y a esta capa también se le agregó la función kernel_regularizer con un valor de 0.001.
- Capa de Salida: 5 unidades con función de activación lineal, ya que se trata de un problema de clasificación multiclase.

Resultados:

Test accuracy: 0.97

Clase	**Precisión**	**Recall**	**F1-score**
Arborio	0.93	0.98	0.95
Basmati	0.99	0.98	0.98
Ipsala	1.00	1.00	1.00
Jasmine	0.98	0.99	0.98
Karacadag	0.98	0.93	0.96
Promedio	0.98	0.97	0.97





Modelo 5

El cuarto modelo representa una refinación adicional del modelo anterior, introduciendo ajustes específicos para abordar la tendencia al sobreajuste y mejorar la capacidad del modelo para generalizar datos no vistos.

Arquitectura del quinto Modelo:

- **Capa de Entrada (Input)**: Se mantiene la rescaling y normalización de las imágenes para preparar adecuadamente los datos de entrada.
- **Capa Convolucional:** Se conserva la arquitectura convolucional con 16 filtros y un tamaño de kernel de 3x3. La función de activación ReLU sigue siendo utilizada para capturar patrones más complejos en las imágenes. La regularización L2 con un parámetro de 0.001 se mantiene para controlar el sobreajuste.
- **Capa de Pooling:** Se mantiene el tamaño del pool a 2x2 y los strides a 2 para preservar más detalles durante la fase de pooling.
- **Capa de Aplanado (Flatten):** Se conserva para transformar los datos en un formato adecuado antes de ingresar a las capas densas.
- **Capas Densas (Fully Connected):** Se introduce una segunda capa densa con 100 unidades y activación ReLU para permitir una representación más rica y compleja de las características extraídas.
- **Capa de Dropout:** Se incorpora una capa de Dropout con una tasa del 0.5 para mitigar posibles problemas de overfitting, desactivando aleatoriamente un 50% de las unidades durante el entrenamiento.
- **Capa de Salida**: La capa de salida conserva 5 unidades y no tiene función de activación específica, ya que se trata de un problema de clasificación multiclase.

Test accuracy: 0.98

Clase	Precisión	Recall	F1-score

Arborio	0.97	0.99	0.98
Basmati	0.98	0.96	0.97
Ipsala	1.00	0.99	1.00
Jasmine	0.96	0.97	0.96
Karacadag	0.99	0.98	0.99
Promedio	0.98	0.98	0.98