Evidencia Individual Deep Learning | Clasificación de Imganes de Frutas

Diego Rodríguez

5 de noviembre de 2023

1. Limpieza y procesamiento de datos

En el ámbito del aprendizaje automático y la ciencia de datos, la calidad de los datos es un factor crítico para el éxito de cualquier proyecto. La base de datos **Fruit Recognition**, disponible en la plataforma Kaggle (https://www.kaggle.com/datasets/chrisfilo/fruit-recognition/), es un recurso proporcionado por el usuario Chris Gorgolewski en 2019 en la plataforma en linea para uso abierto. En la esfera del aprendizaje profundo, una base de datos bien curada es el pilar sobre el cual se construyen modelos predictivos robustos. La base de datos de imágenes de frutas a la que nos referiremos en este reporte se erige como un recurso invaluable para aquellos involucrados en la visión por computadora y la clasificación de imágenes mediante algoritmos de deep learning.

La colecta y organización de una base de datos de imágenes de frutas requiere un enfoque meticuloso para asegurar la diversidad y calidad necesarias para el entrenamiento efectivo de modelos de clasificación. Esta base de datos ha sido ensamblada con precisión, constando de miles de imágenes de alta resolución representando una amplia gama de frutas. Cada imagen ha sido cuidadosamente etiquetada para reflejar la categoría de fruta que representa, proporcionando un terreno sólido para el entrenamiento supervisado.

Esta base de datos se presenta como un recurso didáctico ideal para proyectos de clasificación de imágenes. Su estructura y calidad permiten su aplicación no solo en la investigación académica, sino también en el desarrollo de soluciones prácticas de inteligencia artificial en el ámbito de la clasificación de alimentos y la automatización de la selección de calidad en la industria alimentaria.

1.1. Descripción de la Base de Datos

La base de datos está compuesta por un conjunto diversificado de imágenes pertenecientes a 15 categorías de frutas, incluyendo manzanas, plátanos, naranjas, entre otras. Cada categoría está representada por una serie de imágenes en color, las cuales han sido previamente recolectadas y etiquetadas con el fin de proporcionar un conjunto de datos coherente para

el entrenamiento supervisado. La colección totaliza miles de imágenes, ofreciendo así una cantidad adecuada de datos para garantizar la generalización del modelo.

Las imágenes se presentan en formatos uniformes y han sido preprocesadas para garantizar la consistencia a través del conjunto de datos. Esto incluye normalización de tamaño y resolución, lo que facilita el procesamiento por parte de la red neuronal y asegura que las variaciones en la clasificación sean atribuibles a las características distintivas de las frutas, y no a las discrepancias en la presentación de las imágenes. Este conjunto de datos se ha dividido de manera equitativa entre las clases, asegurando que el modelo no desarrolle sesgos hacia categorías más representadas. Además, se han incluido imágenes que presentan las frutas en distintas orientaciones y condiciones de iluminación, aumentando la robustez del modelo frente a variaciones en las imágenes de entrada.

Figura 1: Ejemplo de imagenes de cada fruta

1.2. Transformación de imagenes

Las imágenes se redimensionan a un tamaño uniforme de 150x150 píxeles utilizando la función ‘cv2.resize‘ de la biblioteca OpenCV. Este ajuste de tamaño es crucial por varias razones:

1. **Consistencia de Entrada**: Las redes neuronales requieren que todas las entradas (en este caso, imágenes) tengan la misma forma. Al estandarizar el tamaño de todas las imágenes, garantizamos que la red pueda procesarlas de manera uniforme.

2. **Reducción de Complejidad Computacional**: Las imágenes de mayor resolución contienen más píxeles y, por tanto, más datos. Al reducir el tamaño de la imagen, disminuimos la cantidad de datos que la red necesita procesar, lo que puede reducir significativamente los requisitos de memoria y tiempo de cálculo.

3. **Mejora del Aprendizaje**: En muchos casos, imágenes más pequeñas pueden ayudar a la red a enfocarse en las características esenciales para la clasificación, en lugar de detalles superfluos que podrían estar presentes en imágenes de alta resolución.

4. **Uniformidad de Características**: Al cambiar el tamaño de todas las imágenes a la misma resolución, nos aseguramos de que las características (como bordes, formas y patrones de color) se escalen de manera uniforme, lo que es importante para que el modelo aprenda de manera efectiva.

Después de procesar todas las imágenes y etiquetas, se devuelven dos arrays de NumPy: uno conteniendo las imágenes redimensionadas y otro con las etiquetas correspondientes. En resumen, la transformación de las imágenes al ajustar su tamaño no solo es un paso necesario para crear un conjunto de datos homogéneo que pueda ser procesado por una red neuronal, sino que también optimiza el proceso de aprendizaje y mejora la eficiencia computacional, facilitando así el desarrollo de modelos robustos y escalables.

2. Entrenamiento de modelo

```
def create_model():
    shape_img = (150,150,3)

    model = Sequential()

    model.add(Conv2D(filters=32, kernel_size=(3,3),
                     input_shape=shape_img, activation='relu',
                     padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    model.add(Conv2D(filters=64, kernel_size=(3,3),
                input_shape=shape_img, activation='relu',
                padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=128, kernel_size=(3,3),
                input_shape=shape_img, activation='relu',
                padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())

    model.add(Dense(128))
    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(len(mapper_fruit_names)))
    model.add(Activation('softmax'))

    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',metrics=['accuracy'])

    return model

model = create_model()
hists = []
divisor = 5

start_time = time.time()
X_train, y_train = load_img(cut_df(df,divisor,1))
y_train = to_categorical(y_train)
callbacks = [EarlyStopping(monitor='val_loss',
            patience=30),
            ModelCheckpoint(filepath='best_model.h5',
            monitor='val_loss', save_best_only=True)]
model.fit(X_train, y_train, batch_size=128, epochs=15,
            callbacks=callbacks, validation_split = 0.1, verbose = 1)
hists.append(model.history.history)
```

En la configuración y el entrenamiento de nuestro modelo de clasificación de imágenes de frutas, se han tomado decisiones cruciales respecto a la arquitectura de la red neuronal y los parámetros de entrenamiento basándonos en una serie de pruebas empíricas y conocimientos previos sobre aprendizaje profundo.

La función ‘create_model()‘ define la arquitectura de una red neuronal convolucional (CNN) que ha demostrado un alto rendimiento para la tarea en cuestión. A continuación, se detallan las elecciones específicas y su justificación:

1. **Arquitectura de la Red Neuronal Convolucional (CNN):**
   - Las CNN son idóneas para el procesamiento de imágenes debido a su habilidad para detectar patrones jerárquicos y características locales a través de los filtros convolucionales.
2. **Capas Convolucionales y de Pooling:**
   - Se inicia con 32 filtros en la primera capa convolucional y se incrementa a 64 en la segunda y 128 en la tercera. Esta progresión permite que la red capture características de bajo nivel en las primeras capas y características de más alto nivel en las profundidades mayores.
   - El kernel de tamaño $(3,3)$ es una elección estándar que equilibra el rendimiento y la complejidad computacional.
   - La activación ’relu’ es utilizada por su eficiencia en la propagación de gradientes y por evitar el problema de las neuronas "muertas"que pueden surgir con la función sigmoidea.
   - El ’padding’ igual a ’same’ garantiza que el tamaño de salida de la convolución sea el mismo que el tamaño de entrada, manteniendo así la información en los bordes de las imágenes.
   - MaxPooling con una ventana de $(2,2)$ reduce la dimensionalidad espacial y ayuda a hacer el modelo más robusto a las variaciones en la posición de las características en las imágenes.
3. **Aplanado y Capas Densas:**
   - La operación de `Flatten` transforma la matriz 3D de activaciones de las capas convolucionales en un vector 1D, permitiendo que los datos se pasen a capas densamente conectadas (Dense layers).
   - Una capa densa con 128 unidades proporciona una capacidad significativa para aprender patrones complejos. Se utiliza la función de activación ’relu’ aquí también.
   - Una capa densa con 256 unidades proporciona una capacidad significativa para aprender patrones complejos. Se utiliza la función de activación ’relu’ aquí también.

- El `Dropout` de 0.5 ayuda a prevenir el sobreajuste reduciendo la co-dependencia entre las neuronas; durante el entrenamiento, el 50 % de las unidades de la capa se .ªpagan.ªleatoriamente en cada paso.

4. **Capa de Salida:**
   - La última capa densa tiene una unidad por cada clase de fruta en el conjunto de datos, con una activación 'softmax' para generar una distribución de probabilidad sobre las clases.
5. **Compilación del Modelo:**
   - Se utiliza la función de pérdida 'categorical_crossentropy' que es adecuada para tareas de clasificación multiclase.
   - El optimizador 'adam' es una elección común por su eficacia general y su adaptabilidad en la tasa de aprendizaje.

El modelo se entrena con un tamaño de lote ('batch_size') de 128 imágenes, lo cual es un balance entre eficiencia computacional y capacidad de generalización. Se optó por 15 épocas para el entrenamiento inicial basándonos en observaciones de la convergencia del accuracy y la pérdida en pruebas anteriores. El uso de 'callbacks' tales como 'EarlyStopping' y 'ModelCheckpoint' permiten una mayor eficiencia en el entrenamiento, deteniéndolo si no se observan mejoras y guardando el mejor modelo encontrado. La división de validación del 10 % durante el entrenamiento proporciona una estimación honesta del rendimiento del modelo en datos no vistos, ayudando a monitorizar y evitar el sobre ajuste.

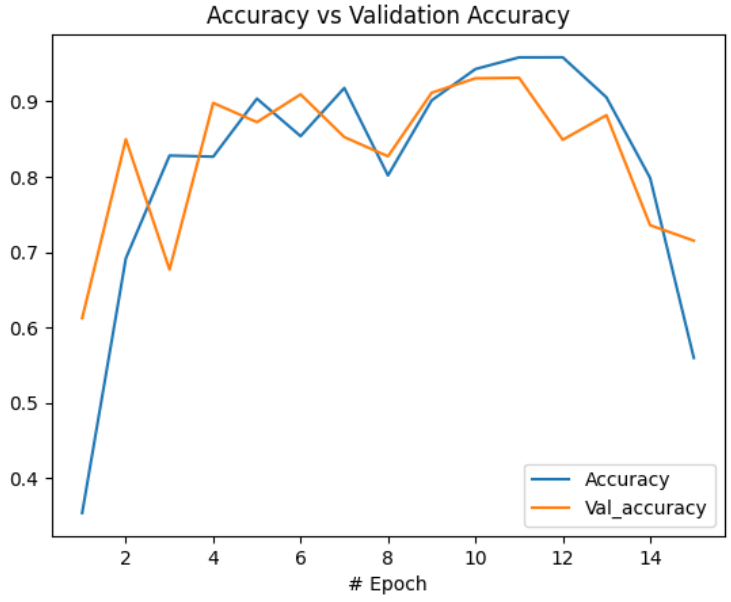

Figura 2: Accuracy vs Validation Accuracy del modelo 1

3. Optimización y refinamiento del modelo

En el proceso de optimización y refinamiento de nuestro modelo de red neuronal convolucional, se realizaron diversas pruebas experimentales para mejorar su rendimiento en la tarea de clasificación de imágenes. Tras un análisis detallado de los resultados preliminares, tomando en cuenta la decaída del rendimiento de la red mostrado en la imagen de accuracy vs validation accuracy a partir de la epoch 10, se implementaron cambios significativos en la arquitectura y los hiperparámetros del modelo. Se eliminó la capa convolucional con 128 filtros y tamaño de kernel de 3×3. Además, se eliminó también la capa densa de 128 unidades para reducirl la complejidad al modelo ya que comenzó a overfitearse. En términos de entrenamiento, ajustamos la paciencia del callback de EarlyStopping a 20 para reducir la tolerancia antes de detener el entrenamiento ya que comenzó a empeorar mucho a lo largo de las epochs, y reducimos el número total de épocas a 10.

4. Configuración y entrenamiento del segundo modelo

```
def create_model():
    shape_img = (150,150,3)

    model = Sequential()

    model.add(Conv2D(filters=32, kernel_size=(3,3),
            input_shape=shape_img, activation='relu',
            padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=64, kernel_size=(3,3),
            input_shape=shape_img, activation='relu',
            padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())

    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(len(mapper_fruit_names)))
    model.add(Activation('softmax'))
```

```
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',metrics=['accuracy'])

    return model

model = create_model()
hists = []
divisor = 5

X_train, y_train = load_img(cut_df(df,divisor,1))
y_train = to_categorical(y_train)

callbacks = [EarlyStopping(monitor='val_loss', patience=20),
             ModelCheckpoint(filepath='best_model.h5',
             monitor='val_loss', save_best_only=True)]

model.fit(X_train, y_train, batch_size=128, epochs=10,
          callbacks=callbacks, validation_split = 0.1,
          verbose = 1)
hists.append(model.history.history)
```

1. **Capas Convolucionales y de Pooling:**
    - Se inicia con 32 filtros en la primera capa convolucional y se incrementa a 64 en la segunda.
    - El kernel de tamaño $(3,3)$ es una elección estándar que equilibra el rendimiento y la complejidad computacional.
    - La activación 'relu' es utilizada por su eficiencia en la propagación de gradientes y por evitar el problema de las neuronas "muertas"que pueden surgir con la función sigmoidea.
    - El 'padding' igual a 'same' garantiza que el tamaño de salida de la convolución sea el mismo que el tamaño de entrada, manteniendo así la información en los bordes de las imágenes.
    - MaxPooling con una ventana de $(2,2)$ reduce la dimensionalidad espacial y ayuda a hacer el modelo más robusto a las variaciones en la posición de las características en las imágenes.
2. **Aplanado y Capas Densas:**
    - La operación de `Flatten` transforma la matriz 3D de activaciones de las capas convolucionales en un vector 1D, permitiendo que los datos se pasen a capas densamente conectadas (Dense layers).

- Una capa densa con 256 unidades proporciona una capacidad significativa para aprender patrones complejos. Se utiliza la función de activación 'relu' aquí también.
- El `Dropout` de 0.5 ayuda a prevenir el sobreajuste reduciendo la co-dependencia entre las neuronas; durante el entrenamiento, el 50% de las unidades de la capa se .ªpagan.ªleatoriamente en cada paso.

3. **Capa de Salida:**
   - La última capa densa tiene una unidad por cada clase de fruta en el conjunto de datos, con una activación 'softmax' para generar una distribución de probabilidad sobre las clases.
4. **Compilación del Modelo:**
   - Se utiliza la función de pérdida 'categorical_crossentropy' que es adecuada para tareas de clasificación multiclase.
   - El optimizador 'adam' es una elección común por su eficacia general y su adaptabilidad en la tasa de aprendizaje.

5. Evaluación y Resultados del segundo modelo

5.1. Resultados

```
### Result of the predictions using 3527 test data ###

Classification Report:
```

	precision	recall	f1-score	support
0	0.81	0.92	0.86	573
1	0.78	0.84	0.81	149
2	0.87	0.90	0.88	107
3	0.97	0.97	0.97	1022
4	0.97	0.94	0.96	411
5	0.91	0.91	0.91	183
6	0.93	0.94	0.94	149
7	0.88	0.78	0.82	148
8	0.89	0.75	0.81	144
9	0.98	0.93	0.95	111
10	0.93	0.90	0.91	97
11	0.98	0.98	0.98	119

```
              12       0.96      0.81      0.88       105
              13       0.95      0.91      0.93       117
              14       0.91      0.85      0.88        92

    accuracy                           0.92      3527
   macro avg       0.91      0.89      0.90      3527
weighted avg       0.92      0.92      0.92      3527


Confusion Matrix:


[[527   1   3  10   1   1   0  11   5   0   2   1   2   3   6]
 [  6 125   6   3   1   3   1   0   2   0   0   0   1   0   1]
 [  1   7  96   2   0   0   0   0   0   0   0   0   1   0   0]
 [ 15   9   2 995   1   0   0   0   0   0   0   0   0   0   0]
 [  9   4   0   7 386   0   0   0   5   0   0   0   0   0   0]
 [  7   3   1   0   2 167   2   0   1   0   0   0   0   0   0]
 [  0   1   0   0   0   5 140   0   0   2   0   0   0   1   0]
 [ 24   1   0   0   1   0   0 115   1   0   5   0   0   0   1]
 [ 26   4   0   2   1   1   0   2 108   0   0   0   0   0   0]
 [  2   0   0   0   0   0   6   0   0 103   0   0   0   0   0]
 [  8   0   0   0   0   0   0   0   0   0  87   1   0   1   0]
 [  1   0   0   0   0   0   0   1   0   0   0 117   0   0   0]
 [  2   4   2   3   0   6   1   1   0   0   0   0  85   1   0]
 [ 10   0   0   0   0   0   0   0   0   0   0   0   0 107   0]
 [  9   1   0   0   3   0   0   1   0   0   0   0   0   0  78]]


# Accuracy: 0.91749
```

5.2. Accuracy del Modelo

El modelo fue evaluado utilizando un conjunto de datos de prueba compuesto por 3527 muestras. Logró una precisión general (‘accuracy‘) del 91.749 %. Este rendimiento refleja la capacidad del modelo para clasificar correctamente las imágenes del conjunto de datos de prueba en las 15 clases distintas de frutas.

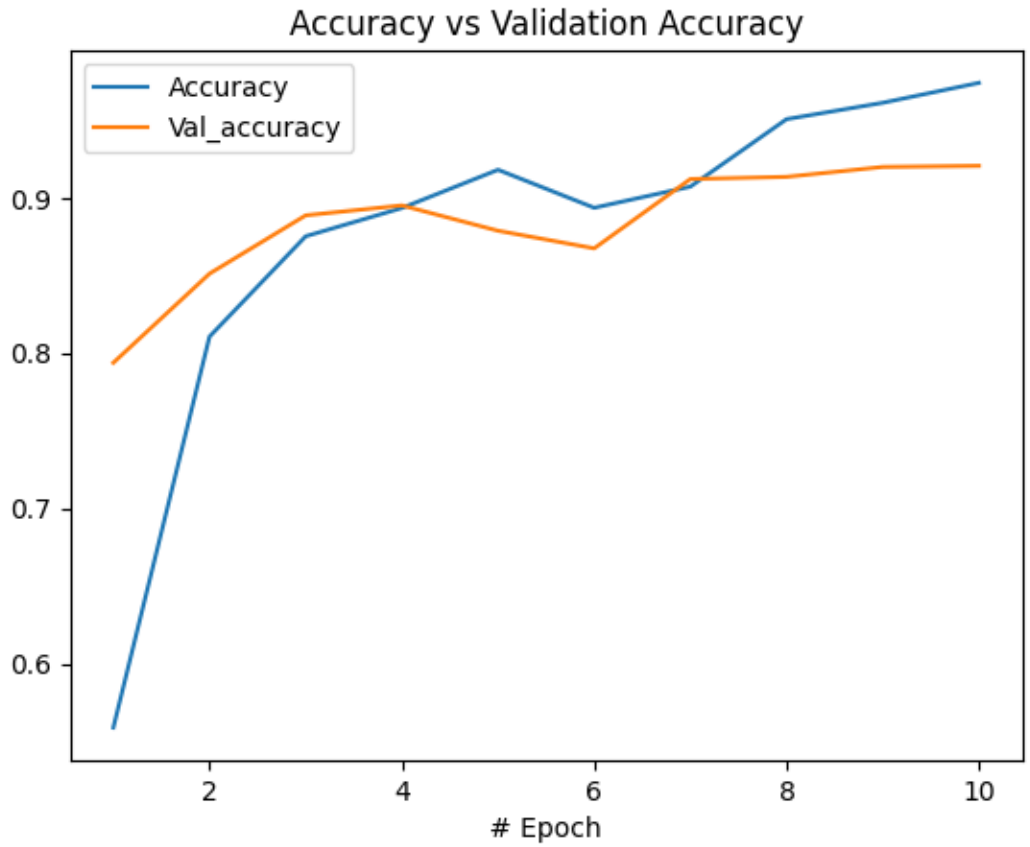

Figura 3: Accuracy vs Validation Accuracy

5.3. Análisis de Precisión y Validación

La gráfica de accuracy frente al accuracy de validación muestra cómo el modelo se comportó durante las fases de entrenamiento y validación a lo largo de 10 épocas. Podemos observar que el accuracy de entrenamiento y validación aumentó consistentemente a lo largo del tiempo, con el accuracy de entrenamiento comenzando alrededor del 70 % y mejorando hasta aproximadamente el 92 %, y la accuracy de validación siguiendo un patrón similar. Esta convergencia de las curvas sugiere que el modelo no está sobreajustado y generaliza bien a nuevos datos.

5.4. Reporte de Clasificación

El reporte de clasificación proporciona una visión detallada del rendimiento del modelo en cada una de las clases. La mayoría de las clases mostraron una alta ‘precisión‘ y ‘recall‘, con puntuaciones de ‘F1-score‘ reflejando un equilibrio entre estas dos métricas. Las clases con menos muestras, como las que corresponden a las etiquetas 2 y 10, aún lograron un alto rendimiento, lo que indica que el modelo es capaz de aprender características distintivas incluso de clases menos representadas.

Las clases que tienen un ‘recall‘ más bajo indican casos donde el modelo tuvo más falsos negativos, mientras que aquellas con una precisión más baja tuvieron más falsos positivos. Por ejemplo, la clase correspondiente a la etiqueta 8 tuvo un recall del 75 %, lo que sugiere que hay margen de mejora para identificar correctamente todas las muestras relevantes de esa clase.

5.5. Matriz de Confusión

La matriz de confusión revela cómo las diferentes clases fueron confundidas por el modelo. La diagonal principal, que muestra el número de predicciones correctas, es significativamente alta para todas las clases, lo que es indicativo de un buen rendimiento general. Las clases fuera de la diagonal principal representan confusiones específicas entre clases, que son relativamente bajas, lo que muestra que el modelo tiene una buena capacidad discriminativa entre las diferentes frutas.

Las filas y columnas de la matriz pueden ser desglosadas para entender mejor las confusiones específicas. Por ejemplo, la clase con la etiqueta 7 fue confundida con la clase con la etiqueta 0 en 24 ocasiones, lo que podría indicar características visuales similares entre estas dos clases que podrían ser examinadas más detenidamente para futuras mejoras del modelo.

En conclusión, el modelo ha demostrado una robusta capacidad de clasificación en el conjunto de datos de prueba. Con un accuracy de más del 90 %, demuestra ser adecuado para la tarea de clasificación de imágenes de frutas. No obstante, ciertas clases podrían beneficiarse de una investigación adicional y de un ajuste fino del modelo para resolver las confusiones específicas y mejorar aún más el accuracy del modelo.

5.6. Prueba con imagen propia

A continuación se realizó pruebas con fotografías capturadas en mi propio celular en la cocina de mi casa.

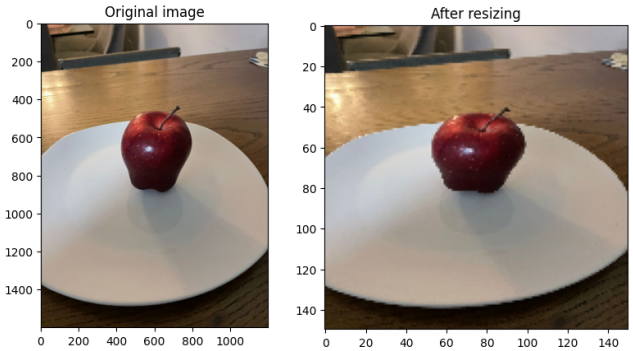

Figura 4: Foto de Manzana original y con redimensionamiento

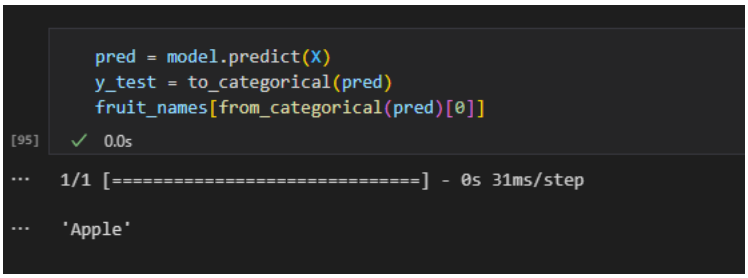

Figura 5: Predicción del modelo

Como se puede observar, el modelo predijo correctamente la fruta mostrada en la foto capturada que no pertenece al dataset.

6. Resumen Final

En resumen, el modelo de Deep Learning desarrollado para la clasificación de imágenes de frutas ha evidenciado una alta eficacia, alcanzando un 'accuracy' global del 91.749 % en el conjunto de datos de prueba. La arquitectura del modelo, consistiendo en capas convolucionales, capas de pooling, una capa de aplanado y capas densas, junto con técnicas de regularización como el Dropout, han permitido que el sistema aprenda y generalice patrones complejos de manera efectiva.

El uso de funciones de activación 'relu', la técnica de MaxPooling y la elección de un optimizador 'adam' demuestran ser decisiones adecuadas que contribuyen a la robustez del modelo. La matriz de confusión y el reporte de clasificación detallado reflejan un alto rendimiento en la mayoría de las clases, aunque se identifican áreas específicas donde el modelo podría mejorar, como en las clases con menores ejemplos o donde la confusión entre clases es más significativa.

La convergencia observada entre el 'accuracy' de entrenamiento y validación sugiere que no hay un sobreajuste significativo y que el modelo es capaz de generalizar a nuevos datos, lo cual se ve respaldado por la prueba exitosa con una imagen capturada desde un celular, que no pertenecía al conjunto de datos original, y que aún así fue clasificada correctamente. Por lo tanto, este modelo se perfila como una herramienta prometedora para la clasificación automática de imágenes de frutas, con aplicaciones potenciales en áreas como el control de calidad alimentario, comercio electrónico y educación nutricional. Para futuras iteraciones, sería beneficioso aumentar la diversidad y cantidad de datos de entrenamiento, especialmente en aquellas clases donde el rendimiento fue menor, y explorar técnicas adicionales de aumento de datos y ajuste fino para refinar la precisión y el 'recall' del modelo.

Este proyecto ilustra el potencial del aprendizaje profundo en aplicaciones de visión por computadora y establece una base sólida para futuras investigaciones y mejoras en la clasificación de imágenes de frutas y otros objetos similares.