Clasificación de frutas frescas y podridas mediante...

Calcina Huaranga Nestor Giraldo, Huaman Meza Esai Josue
Reyes Robles Alan Jerry, Zegarra Guardamino Jorge Luis
Universidad Nacional de San Agustín
ncalcinahu@unsa.edu.pe, ehuamanm@unsa.edu.pe
areyesr@unsa.edu.pe, jzegarragu@unsa.edu.pe
Arequipa - Perú

20/05/2023

Resumen

In recent decades, fruit production in Peru has experienced significant growth due to the increasing global demand for fresh and healthy fruits. The country has developed a strong fruit export industry, shipping its products to international markets. However, sometimes fruits can rot during transportation or at the point of consumption, which has led to the need to develop technologies to identify fresh and rotten fruits. In this context, a fully connected neural network has been developed in Python to classify fresh and rotten fruits. This research highlights the growing adoption of artificial intelligence and machine learning in the fruit production industry and how these technologies can help improve product quality and safety.

1. Introducción

En la agricultura específicamente en la producción de frutas, es muy importante controlar la calidad del producto final, las frutas que van a ser puestas a disposición del cliente final deben estar frescas, tener un color uniforme, brillante, textura firme, en general estar aptas para el consumo humano. La calidad de la fruta puede verse afectada durante la producción y/o traslado hacia los centros de distribución, ¿cómo podemos saber si una fruta está fresca o no?. En este trabajo proponemos un modelo de red neuronal convolucional para la clasificación de frutas frescas y en mal estado haciendo uso de un dataset de 13599 imágenes de manzana, plátano y naranja etiquetadas como frescas y podridas.

2. Trabajos Relacionados

Se desarrolló un sistema de visión artificial para la detección de defectos de la piel de la fruta en el estudio [1]. El color es la principal característica para la categorización y se ha utilizado un algoritmo de aprendizaje de redes neuronales densas (fully conected). Las redes neuronales densas pueden manejar datos de entrada de cualquier tamaño y forma, ya que todas las unidades de una capa están conectadas con las unidades de la capa anterior. La precisión en la clasificación mediante aprendizaje automático se basa principalmente en las características extraídas y las características que se eligen para pasar al algoritmo de aprendizaje automático.

3. Materiales y Métodos

3.1. Dataset, Adquisición y preprocesamiento

El conjunto de datos fue obtenido de kaggle; "Fruits fresh and rotten for classification" y está compuesto por 13599 imágenes; 5904 de frutas frescas y 7695 de frutas malogradas o podridas entre manzana, plátano y naranja. El 80 % de los datos fue utilizado para entrenamiento y el 20 % para pruebas.

FRUTA	FRESH	ROTTEN	TOTAL
Apples	1693	2342	4035
Bananas	1581	2224	3805
Oranges	1466	1595	3061
Total	4740	6161	10901

Cuadro 1: Dataset de Entrenamiento

FRUTA	FRESH	ROTTEN	TOTAL
Apples	395	601	996
Bananas	381	530	911
Oranges	388	403	791
Total	1164	1534	2698

Cuadro 2: Dataset de Prueba

- El tamaño original de las imágenes fue de 100 x 100 pixeles, luego fue redimensionada a 80x80x3 con numpy y posteriormente convertida a escala de grises, finalmente escalamos los datos a un rango de 0 a 1 con MinMaxScaler de sklearn y convertimos las etiquetas de las imágenes a una representación one-hot-encoding con to_categorical de tensorflow.

3.2. Redes neuronales densas

Las redes neuronales densas, también conocidas como fully connected neural networks o redes neuronales completamente conectadas, son un tipo de arquitectura de redes neuronales en el que cada neurona en una capa está conectada a todas las neuronas de la capa siguiente. Esto significa que todas las unidades de una capa están conectadas a todas las unidades de la capa siguiente, lo que genera una red densamente conectada.

En una red neuronal densa, los datos fluyen en una dirección desde la capa de entrada hasta la capa de salida a través de múltiples capas ocultas. Cada unidad en una capa oculta toma como entrada todas las salidas de las unidades en la capa anterior y realiza una transformación lineal seguida de una función de activación no lineal.

Algunas de sus características y ventajas son las siguientes:

- Flexibilidad en la entrada: Manejan datos de entrada de cualquier tamaño y forma, ya que todas las unidades en una capa están conectadas a todas las unidades en la capa siguiente.
- Capacidad para aprender representaciones complejas: Tienen la capacidad de aprender representaciones jerárquicas y complejas de los datos.
- Entrenamiento eficiente: Pueden entrenarse de manera eficiente en grandes conjuntos de datos.
- Representación compacta de los datos: Pueden aprender a extraer características importantes de los datos y representarlos de manera más eficiente.

3.3. Modelo propuesto para la clasificación de frutas frescas y podridas

En este trabajo utilizamos Keras Model Sequential de tensorflow con 4 capas de 1024, 256, 64 y 16 neuronas respectivamente, todas con activación ReLU. Para la parte de optimización utilizamos Adam con un learning rate de 0.003 sin validación cruzada, se entrenó inicialmente con 10 épocas y batch_size de 32.

- Total params: 6,834,546
- Trainable params: 6,834,546
- Non-trainable params: 0
- El tamaño de la imagen de entrada es 80x80 en escala de grises y la salida dos valores 0 para frutas frescas y 1 para frutas podridas.

Layer (type)	Output Shape	Param
dense_35 (Dense)	(None, 1024)	6554624
dense_36 (Dense)	(None, 256)	262400
dense_37 (Dense)	(None, 64)	16448
dense_38 (Dense)	(None, 16)	1040
dense_39 (Dense)	(None, 2)	34

Cuadro 3: Tabla de Parámetros: Model: sequential

4. Resultados y Conclusiones

Este es un trabajo desafiante para nosotros, ya que somos muy principiantes para nuestro primer artículo. Aquí están las principales cuestiones que se centran en este documento que se da a continuación.
Para el proyecto usamos el conjunto de datos de frutas frescas y podridas del repositorio de Kaggle. Primero dividimos el conjunto de datos en entrenamiento (80 %) y prueba (20 %). Se utilizó la biblioteca de Python "Keras"para implementar este modelo MPL en Google Colab, que utiliza GPU Tipo T4, 5.8 GB de RAM y 25.30 GB de espacio en disco.

Figura 1: GPU

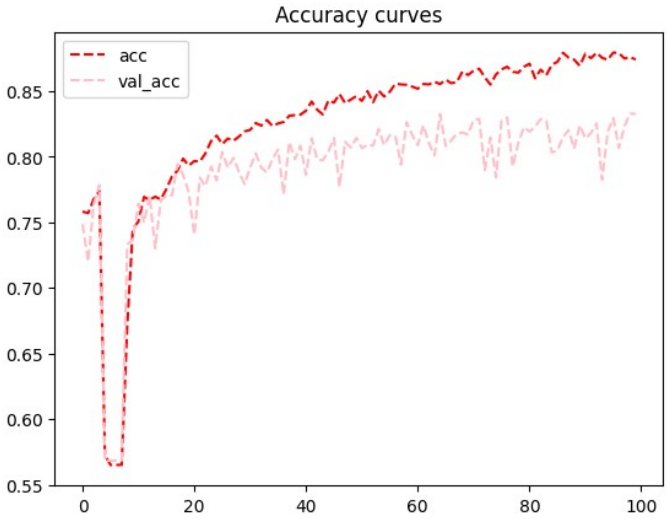


Figura 2: Accuracy Curves

Para el modelo MPL utilizado, se observa el efecto de mayor precisión a medida que se aumentan las épocas de entrenamiento.

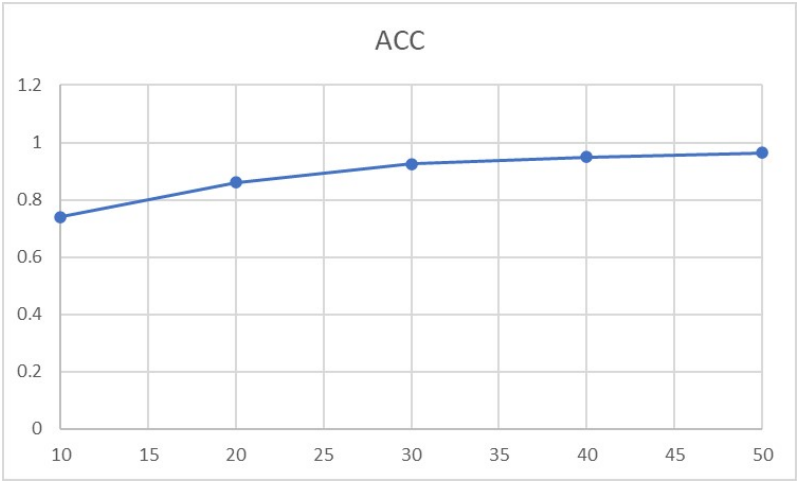


Figura 3: ACC

También se pudo observar que a medida que se aumentaba la cantidad de Bach-Size, la precisión mejoraba. Se verifica que a partir del valor de Bach-Size = 30 la ganancia en precisión se estaciona en resultados aceptables aprox a 95

Los parámetros de optimización se utilizan para optimizar el rendimiento de nuestro modelo al actualizar los parámetros de peso que minimizan nuestra función de pérdida. Nuestro objetivo es reducir la pérdida de nuestra red neuronal mejorando los parámetros de nuestra red. La función de pérdida calcula la pérdida haciendo coincidir el valor real y el valor predicho por una red neuronal. Aquí, en nuestro trabajo, investigamos sobre cuatro optimizadores, y las precisiones se comparan en la Figura 6 y se determina el mejor optimizador.

Los optimizadores revisados son SGD, Adam, Adagrad y RMSprop. El optimizador Adam es el mejor optimizador para nuestro modelo y ofrece una precisión del 97,82 %. Sin embargo, RMSprop también tiene una precisión del 85,64 %. sgd dio una precisión del 34,50 % y Adagrad del 54,75 %.

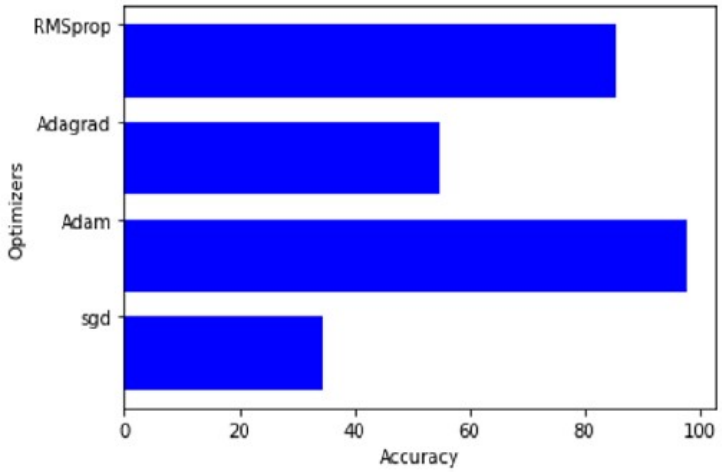


Figura 4: Optimizadores

Durante el entrenamiento de la red neuronal, se actualiza una cierta cantidad de pesos. Estos pesos se denominan tasas de aprendizaje. Este es un hiper parámetro importante utilizado en el modelo CNN cuyo rango está entre 0,0 y 1,0. En nuestro modelo, adoptamos cuatro tasas de aprendizaje y observamos los efectos de esas tasas de aprendizaje en la precisión. Las cuatro tasas de aprendizaje son 0,1, 0,01, 0,001 y 0,0001. De la Figura 7 se observa que al disminuir las tasas de aprendizaje de 0.1 a 0.0001 se mejora la Figura 4.Efectos del tamaño del lote precisión de 17.36 % a 97.82 %.

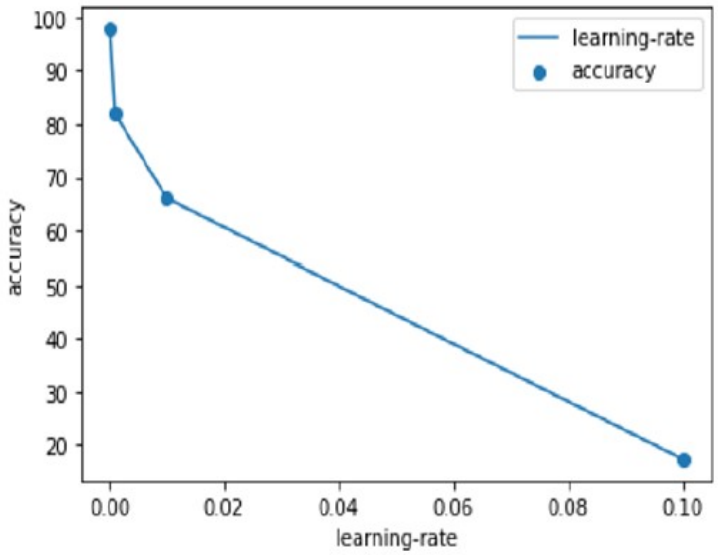


Figura 5: Aprendizaje

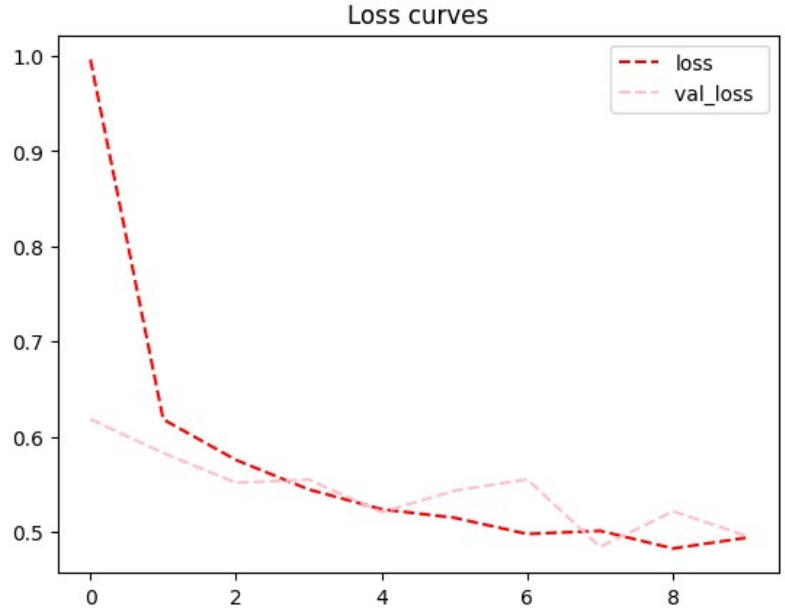
Para nuestro modelo optamos por el hiper parámetro learning-rate = 0.003.

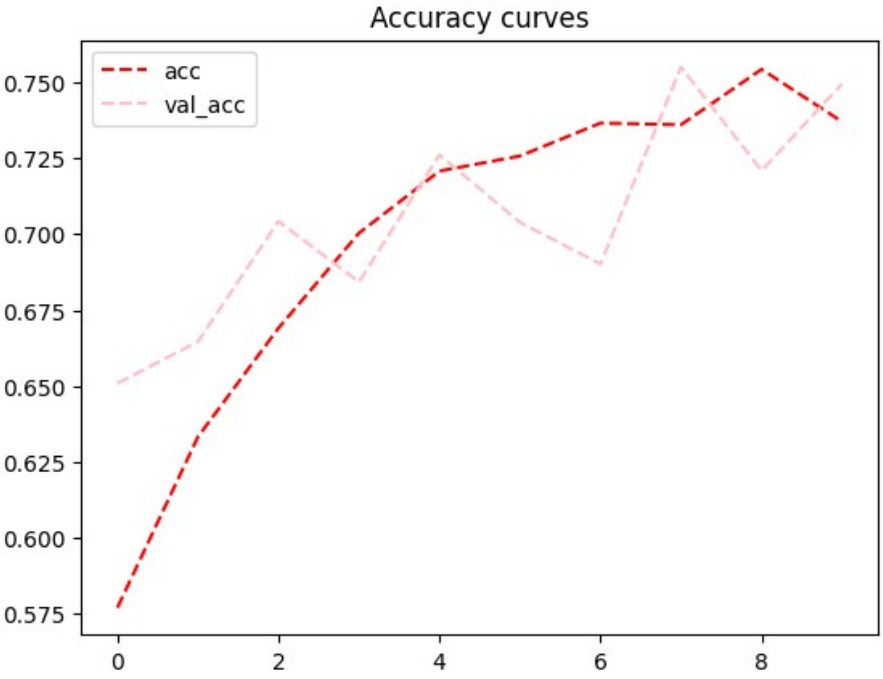
```
[ ]  from tensorflow.keras.optimizers import Adam

     model.compile(
       optimizer=Adam(learning_rate=0.003),
       loss='categorical_crossentropy',
       metrics=['accuracy'],
     )
```

Figura 6: Hiper Parámetro

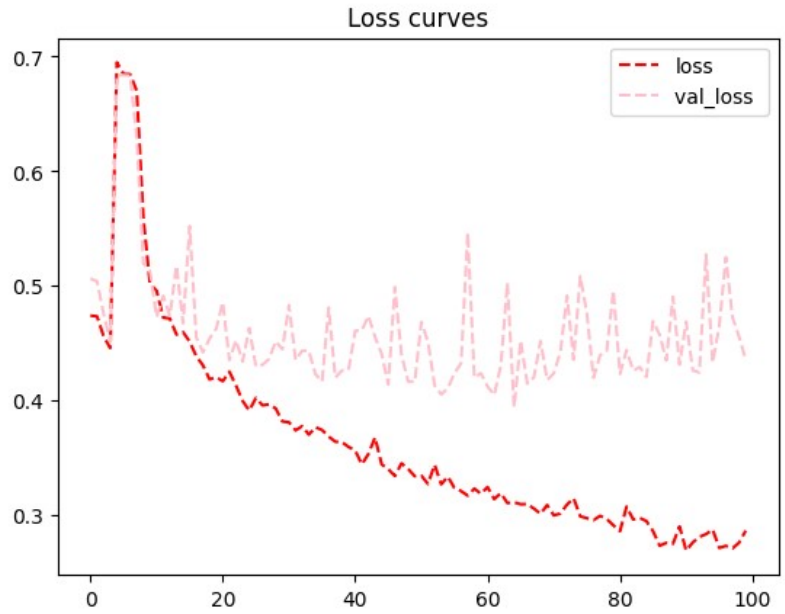
5. Pruebas

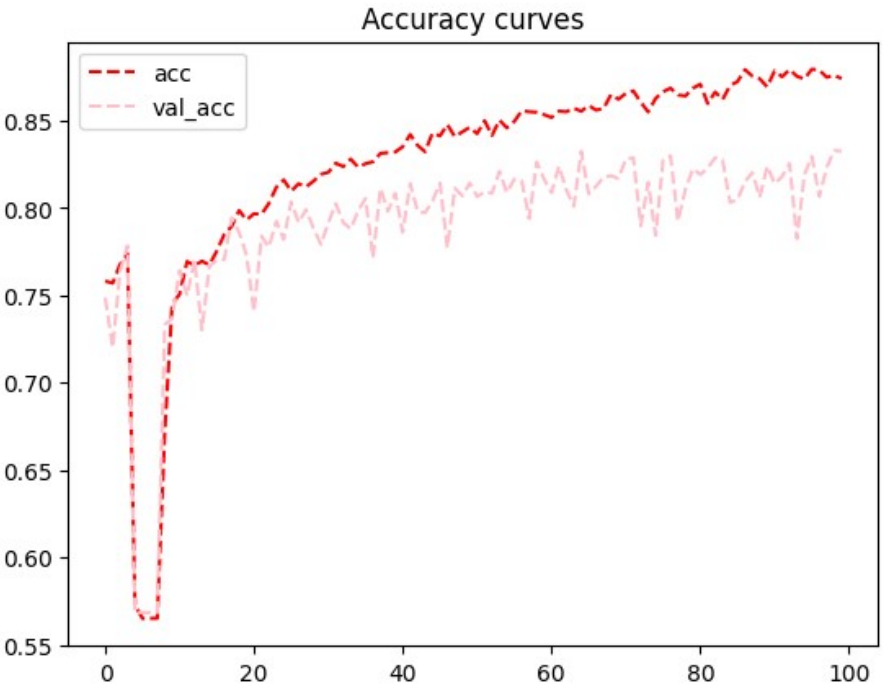
- Epochs = 10, Bach Size = 32



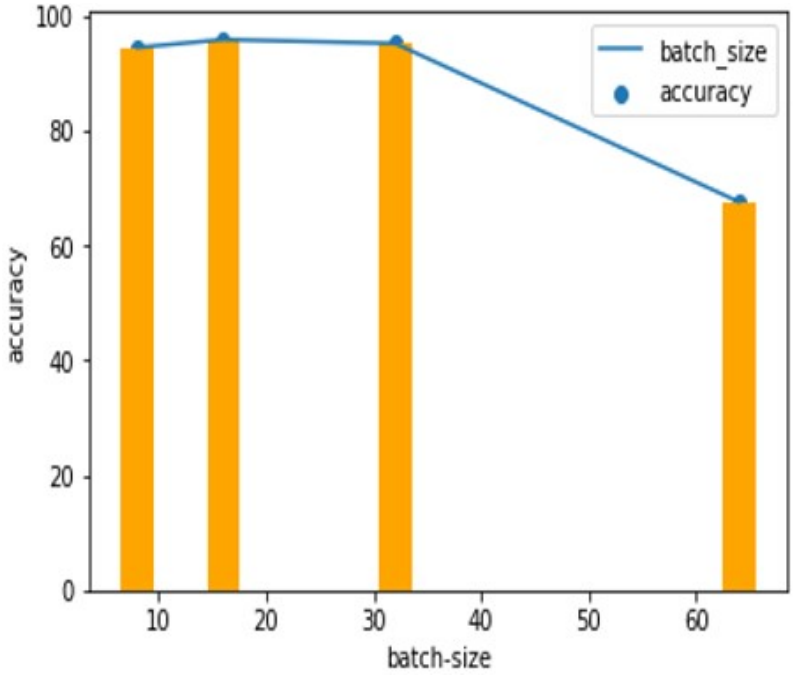


- Epochs = 100, Bach Size = 32





- Probando el Modelo con 100 épocas:



Referencias

[1] L. A. T. X. Wang, L. *Detection of fruit skin defects using machine vision system. In 2013 Sixth International Conference on Business Intelligence and Financial Engineering.* EEUU, Boston, MA, 2013. pp. 44-48.