TAREA 9: CLASIFICACIÓN DE TOMATE MEDIANTE REDES NEURONALES CONVOLUCIONALES.

Marcos Rafael Roman Salgado[1]

[1]*UACh, Texcoco de Mora – <marcosrafael2000@hotmail.com>*

RESUMEN

Este reporte describe el desarrollo de un clasificador de tomates utilizando Redes Neuronales Convolucionales (CNN) para detectar la madurez. La detección precisa de la madurez es esencial para la clasificación, el momento de cosecha y el control de calidad en la agricultura. Este estudio utiliza CNN para automatizar la clasificación de tomates según su madurez, lo que afecta el valor en el mercado y la vida útil. Entrenado con un conjunto de imágenes de tomates categorizadas por madurez, el clasificador muestra la efectividad de las CNN en mejorar la eficiencia y precisión de la clasificación. Los resultados sugieren que las CNN pueden optimizar la gestión de la producción de tomates mediante una detección de madurez automatizada y confiable.

Palabras-clave: *Clasificación de tomates, Redes Neuronales Convolucionales, Detección de madurez, Tecnología agrícola, Procesamiento de imágenes*

ABSTRACT

This report describes the development of a tomato classifier using Convolutional Neural Networks (CNNs) for detecting ripeness. Accurate ripeness detection is crucial for classification, harvest timing, and quality control in agriculture. This study uses CNNs to automate the classification of tomatoes based on their maturity, impacting market value and shelf life. Trained on a dataset of tomato images categorized by ripeness, the classifier demonstrates the effectiveness of CNNs in enhancing classification efficiency and accuracy. The findings suggest that CNNs can significantly improve tomato production management and optimization by providing reliable, automated ripeness detection.

Keywords: *Tomato classification, Convolutional Neural Networks, Ripeness detection, Agricultural technology, Image processing*

1. INTRODUCCIÓN

La detección del nivel de madurez de los tomates es un aspecto crítico en la cadena de producción y comercialización agrícola. La madurez de los tomates afecta no solo su sabor y calidad, sino también su valor en el mercado y su vida útil en el punto de venta. Una clasificación precisa permite a los agricultores y distribuidores tomar decisiones informadas sobre el momento de la cosecha y el proceso de comercialización, optimizando así los recursos y reduciendo el desperdicio.

Las técnicas tradicionales de clasificación de tomates a menudo dependen de la inspección visual humana, lo cual es subjetivo y puede ser propenso a errores. Con el avance de la tecnología, se han desarrollado métodos automatizados para mejorar la precisión y eficiencia en este proceso. En este contexto, las Redes Neuronales Convolucionales (CNN) han demostrado ser herramientas poderosas para el procesamiento y análisis de imágenes, proporcionando una solución eficaz para la detección de la madurez de los tomates.

Figura 1. Diversidad de color en madurez del jitomate.

Fuente: https://proain.com/blogs/notas-tecnicas/claves-en-el-manejo-postcosecha-de-las-frutas-y-hortalizas.

Este estudio se centra en la implementación de un clasificador basado en CNNs para identificar diferentes niveles de madurez en imágenes de toma-

tes. La metodología incluye el preprocesamiento de datos, el diseño y entrenamiento de la red neuronal, y la evaluación del desempeño del modelo. Los resultados obtenidos muestran la capacidad de las CNNs para mejorar la clasificación automatizada de tomates, con implicaciones significativas para la industria agrícola.

Atencion

En esta sección, es importante destacar que la detección de madurez de tomates utilizando redes neuronales convolucionales no solo tiene un impacto en la calidad del producto final, sino también en la sostenibilidad de la producción agrícola. La implementación de tecnologías avanzadas como las CNNs puede reducir el uso de recursos al minimizar el desperdicio de productos que no cumplen con los estándares de madurez.

Además, el uso de técnicas automatizadas para la clasificación puede mejorar la trazabilidad del producto y asegurar una mayor consistencia en la calidad de los tomates distribuidos en el mercado. Esta práctica representa un paso hacia la modernización de la agricultura, haciendo uso de herramientas de inteligencia artificial para enfrentar los desafíos de una industria en constante evolución.

Es fundamental considerar la integración de estas tecnologías en los procesos agrícolas para aprovechar al máximo sus beneficios y contribuir a un futuro más eficiente y sostenible en la producción de alimentos.

2. MATERIALES Y MÉTODOS

2.1. Dataset

Para el desarrollo del clasificador, se utilizó un conjunto de datos de imágenes de tomates proporcionado por el profesor. Las imágenes estaban clasificadas en diferentes categorías de madurez: verde, quebrado, rayado, naranja, rojo-naranja y rojo. Las imágenes fueron recopiladas bajo condiciones de iluminación controladas y desde diversas perspectivas para asegurar una representación variada de cada categoría. Inicialmente, el dataset se dividió en tres conjuntos:

- Entrenamiento con 400 fotos por clase (57 %)
- Validación con 100 fotos por clase (14 %)
- Prueba con 200 fotos por clase (28 %)

Tras realizar varias pruebas y ajustes, y tras recibir la confirmacion del profesor que era valido realizar cambios en los datasets, se modificaron los datasets en los experimentos a un total de:

- 550 imágenes de entrenamiento por clase.
- 100 de prueba por clase.
- 50 de validación por clase.

2.2. Preprocesamiento

Las imágenes fueron preprocesadas utilizando el módulo `ImageDataGenerator` de TensorFlow/Keras con las siguientes adaptaciones para el entrenamiento:

- **Escalado:** Las imágenes fueron escaladas a un rango de [0, 1] mediante `rescale=1./255`.
- **Aumento de datos:** Se aplicaron técnicas de aumento de datos para mejorar la robustez del modelo y prevenir el sobreajuste, incluyendo:
  - Rotación aleatoria (hasta 20 grados).
  - Desplazamiento horizontal y vertical (hasta 20 %).
  - Corte (shear) aleatorio (hasta 20 %).
  - Zoom aleatorio (hasta 20 %).
  - Flip horizontal.
  - Rango de brillo ajustable (0.5 a 1.5).
- **Generadores de datos:** Se utilizaron generadores para cargar las imágenes desde directorios y aplicar las transformaciones especificadas:

Código 1. Configuracion de preprocesamiento

```
train_datagen =
    ImageDataGenerator(
     rescale=1./255,
     rotation_range=20,
     width_shift_range=0.2,
     height_shift_range=0.2,
     shear_range=0.2,
     zoom_range=0.2,
     horizontal_flip=True,
     fill_mode='nearest',
     brightness_range=[0.5, 1.5]
)

validation_datagen =
    ImageDataGenerator(rescale
    =1./255)
test_datagen =
    ImageDataGenerator(rescale
    =1./255)
```

```
15
16 batch_size = 90
17 target_size = (256, 256)
```

2.3. Modelo

El modelo de red neuronal convolucional (CNN) se diseñó utilizando la arquitectura VGG19 preentrenada y se ajustó con las siguientes especificaciones:

2.4. Modelo

El modelo de red neuronal convolucional (CNN) se diseñó utilizando la arquitectura VGG19 preentrenada y se ajustó con las siguientes especificaciones:

- **Modelo Preentrenado:** Se utilizó VGG19 con pesos preentrenados en ImageNet. La arquitectura fue ajustada para no entrenar las primeras capas, manteniendo entrenables solo las capas especificadas ('block5$_conv4$').

  El código para configurar el modelo preentrenado es el siguiente:

Código 2. Configuración de modelo preentrenado

```
1  from tensorflow.keras.applications
       import VGG19
2
3  pre_trained_model = VGG19(
       input_shape=(150,150,3),
4
       include_top=False,
5                              weights='
       imagenet')
6
7  pre_trained_model.trainable = True
8
9  layers_to_train = ['block5_conv4']
10
11 for layer in pre_trained_model.
       layers:
12     if layer.name in
       layers_to_train:
13         layer.trainable = True
14     else:
15         layer.trainable = False
```

- **Arquitectura del Modelo:** Se añadió una capa de flattening seguida de varias capas densas con normalización por lotes (Batch Normalization), dropout y regularización L2.

El código para configurar la arquitectura del modelo es el siguiente:

Código 3. Configuración de capas en el modelo

```
1  from tensorflow.keras import Model,
       layers, Sequential,
       regularizers
2
3  model = Sequential([
4      pre_trained_model,
5      layers.Flatten(),
6
7      layers.Dense(512,
       kernel_regularizer=regularizers.
       l2(0.01)),
8      layers.Activation('relu'),
9      layers.Dropout(0.2),
10
11     layers.Dense(256,
       kernel_regularizer=regularizers.
       l2(0.01)),
12     layers.BatchNormalization(),
13     layers.Activation('relu'),
14     layers.Dropout(0.3),
15
16     layers.Dense(128,
       kernel_regularizer=regularizers.
       l2(0.01)),
17     layers.BatchNormalization(),
18     layers.Activation('relu'),
19     layers.Dropout(0.3),
20
21     layers.Dense(6, activation='
       softmax')
22 ])
```

- **Compilación y Entrenamiento:** El modelo fue entrenado durante 100 épocas con un tamaño de batch de 90 imágenes, se utilizó el optimizador SGD con tasa de aprendizaje adaptativa y la función de pérdida de entropía cruzada categórica. Se aplicaron callbacks para guardar los mejores pesos y detener el entrenamiento temprano si no se observaban mejoras.

El código para compilar y entrenar el modelo es el siguiente:

Código 4. Configuración de Entrenamiento

```
1  from tensorflow.keras.optimizers
       import SGD
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

model.compile(optimizer=SGD(learning_rate=0.01, momentum=0.9, nesterov=True),
              loss='categorical_crossentropy',
              metrics=['acc'])

checkpoint = ModelCheckpoint(filepath='pesos_checkpoint.hdf5', save_best_only=True, verbose=1)
early_stop = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True, mode='min')

history = model.fit(
    train_generator,
    validation_data=validation_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=100,
    validation_steps=validation_steps,
    verbose=2,
    callbacks=[checkpoint, early_stop]
)
```

2.5. Evaluación

El desempeño del clasificador se evaluó utilizando el conjunto de datos de prueba. Se calcularon métricas como la precisión, el recall y la F1-score para cada categoría de madurez. Además, se generó una matriz de confusión para analizar las clasificaciones correctas e incorrectas y evaluar la efectividad general del modelo.

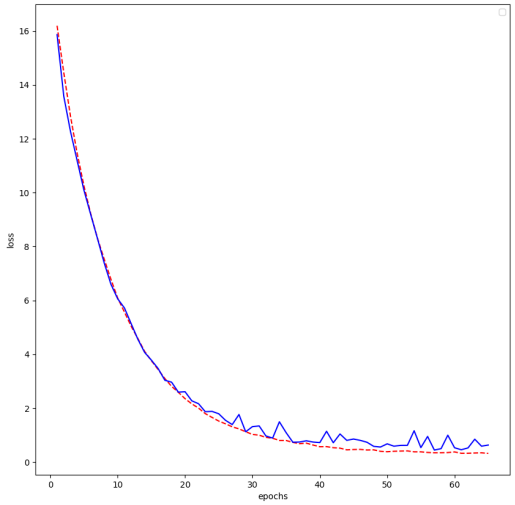
2.6. Herramientas y Entorno

El desarrollo del modelo se realizó utilizando el framework TensorFlow/Keras en Google Colab, que proporciona acceso a recursos de computación acelerada con TPU. Las imágenes fueron procesadas y el modelo entrenado en este entorno para aprovechar su capacidad de procesamiento eficiente.

3. RESULTADOS

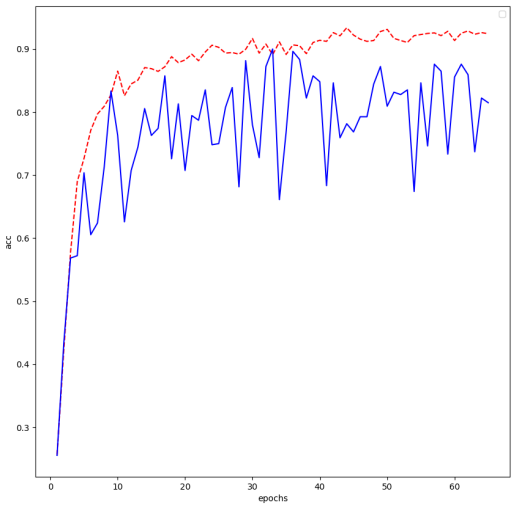
El desempeño del clasificador se evaluó utilizando el conjunto de datos de prueba. Se calcularon métricas como la precisión, el recall y la F1-score para cada categoría de madurez. Además, se generó una matriz de confusión para analizar las clasificaciones correctas e incorrectas y evaluar la efectividad general del modelo.

Figura 2. Resultados de entrenamiento(Loss)



Fuente: autoría propia.

Figura 3. Resultados de entrenamiento(Precision). (MNIST)



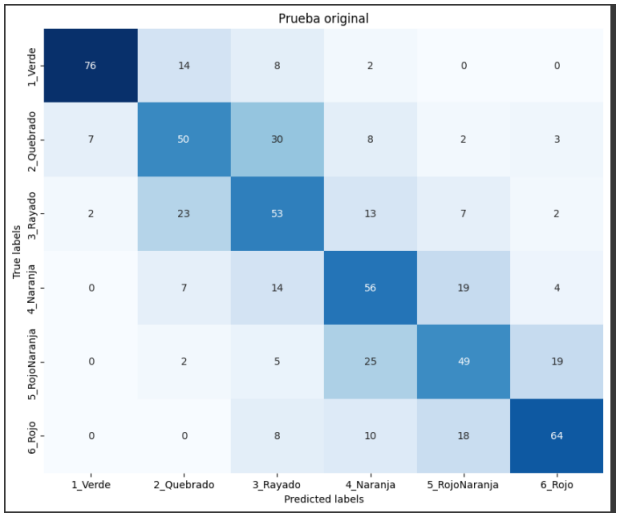
Fuente: autoría propia.

Figura 4. Reporte de métricas

	precision	recall	f1-score	support
1_Verde	0.89	0.76	0.82	100
2_Quebrado	0.58	0.50	0.54	100
3_Rayado	0.45	0.53	0.49	100
4_Naranja	0.46	0.56	0.51	100
5_RojoNaranja	0.50	0.49	0.49	100
6_Rojo	0.70	0.64	0.67	100
accuracy			0.58	600
macro avg	0.60	0.58	0.59	600
weighted avg	0.60	0.58	0.59	600

Fuente: autoría propia.

Figura 5. Mapa de calor



Fuente: autoría propia.

4. DISCUSIÓN

El modelo de red neuronal convolucional (CNN) utilizado en esta práctica fue el VGG19 preentrenado, ajustado para la clasificación de la madurez del tomate en seis categorías diferentes. A pesar de haber aplicado técnicas de data augmentation y ajustes en las capas densas, el modelo presentó un desempeño moderado con una precisión general alrededor del 55 %. Varias observaciones se destacan en la discusión de estos resultados:

- **Errores y Confusiones Entre Clases:** Las confusiones más comunes ocurrieron entre las clases 6 (Rojo) y 5 (RojoNaranja), clases 5 (RojoNaranja) y 4 (Naranja), así como en las clases 2 (Quebrado) y 3 (Rayado). Esto sugiere que las diferencias visuales entre estas etapas de madurez son sutiles y difíciles de distinguir, incluso para el modelo planteado. La clase 1 (Verde) mostró menos confusiones en comparación con otras clases, indicando que las características visuales de los tomates verdes son más distintivas.
- **Impacto de la Data Augmentation:** La aplicación de técnicas de data augmentation, como rotaciones, desplazamientos y ajustes de brillo, ayudó a mejorar la robustez del modelo al hacer que el modelo sea más adaptable a variaciones en las imágenes. Sin embargo, estos aumentos no fueron suficientes para llevar la precisión a niveles más altos.
- **Ajuste de Capas del Modelo Preentrenado:** Mantener entrenables solo las capas más profundas del VGG19 permitió aprovechar el conocimiento previo del modelo sin sobreajustar las primeras capas a este conjunto de datos específico. No obstante, podría ser beneficioso explorar diferentes combinaciones de capas entrenables para encontrar un equilibrio óptimo.
- **Tamaño del Conjunto de Datos:** El conjunto de datos, aunque representativo, puede no ser lo suficientemente amplio para capturar toda la variabilidad en la apariencia de los tomates en diferentes condiciones. Ampliar el conjunto de datos con más imágenes y bajo diferentes condiciones de iluminación podría mejorar el rendimiento del modelo.

5. CONCLUSIONES

En esta práctica, se desarrolló un clasificador de la madurez del tomate utilizando un modelo CNN preentrenado (VGG19) y se aplicaron diversas técnicas de procesamiento y ajuste del modelo. Aunque el rendimiento alcanzado fue moderado, con una precisión general alrededor del 58 %, se identificaron áreas clave para mejorar:

- **Ampliación y Diversificación del Conjunto de Datos:** Recopilar más datos y aumentar la variedad en las condiciones de captura de las imágenes podría ayudar a mejorar la precisión del modelo.
- **Ajustes en la Arquitectura del Modelo:** Experimentar con diferentes capas entrenables y arquitecturas puede encontrar un mejor balance entre sobreajuste y generalización.
- **Optimización de Parámetros:** Probar con diferentes hiperparámetros, como la tasa de aprendizaje y el tipo de optimizador, puede llevar a mejores resultados.

Este ejercicio no solo mostró los desafíos en la cla-

sificación de imágenes agrícolas, sino que también resaltó la importancia de la experimentación y la optimización continua en el desarrollo de modelos de aprendizaje automático. La precisión moderada obtenida sugiere que hay margen para mejoras significativas y futuras investigaciones en este campo.

APÉNDICES

Bitacora de desarrollo

Inicialmente, comencé realizando un primer modelo bastante sencillo con algunas capas convolucionales para revisar cómo se comportaba. Al ver los resultados, que eran muy malos, continué añadiendo complejidad para revisar cómo se comportaban los cambios. Lo primero que noté fue que, a pesar de que el modelo no mejoraba significativamente en términos de precisión, los tiempos de desarrollo sí se aumentaban.

- **Modelo Sencillo:** Primeros intentos con una arquitectura básica compuesta por unas pocas capas convolucionales.
- **Añadiendo Complejidad:** Inclusión de más capas y ajustes para mejorar el rendimiento, como capas de pooling y activación.

Comencé a aplicar diversas técnicas de regularización, como la regularización L2 y el dropout, para evitar el sobreajuste y mejorar la generalización del modelo. Sin embargo, al ser pocas imágenes y no tener una arquitectura adecuada, seguían surgiendo problemas como la baja precisión y el sobreajuste. En este punto, decidí investigar a través de Google Academic, encontrando numerosos artículos que presentaban arquitecturas interesantes pero sumamente complejas. Algunas de estas arquitecturas intenté implementarlas, aplicando las imágenes del dataset. No obstante, los tiempos de ejecución eran muy largos y, en más de una ocasión, por errores ajenos al código, el programa fallaba y se perdía el progreso.

- **Técnicas de Regularización:** Experimentación con regularizaciones sin mejora significativa, pero con aprendizaje valioso sobre la importancia de estas técnicas.
- **Investigación y Nuevas Arquitecturas:** Búsqueda en Google Academic y prueba de arquitecturas complejas como ResNet y DenseNet.

Tras varios intentos y gastar el 50 % de mis unidades de procesamiento con arquitecturas complejas, hice un cambio de paradigma y comencé a utilizar modelos preentrenados, como VGG19 y ResNet, dejando algunas capas para entrenamiento y añadiendo capas extra que se adaptaran a las imágenes presentes. Este enfoque resultó en una mejora notable en los modelos y en la elección final del modelo que tuvo un mejor comportamiento, tanto en precisión como en eficiencia.

- **Cambio de Paradigma:** Uso de modelos preentrenados y ajuste fino de capas, lo cual permitió aprovechar características preaprendidas y reducir el tiempo de entrenamiento.
- **Mejoras y Selección de Modelos:** Resultados más satisfactorios con modelos preentrenados y personalización de la arquitectura para el dataset específico.

Tras varios intentos y mucho estrés, decidí conservar un modelo sencillo como referencia y, posteriormente, tras más estudio y pruebas, buscar modificar el modelo de manera incremental. Adicionalmente, en una reunión del diplomado, escuché que era válido modificar el dataset. Aproveché esta oportunidad para añadir más imágenes a la carpeta de entrenamiento con la intención de mejorar la robustez del modelo y su capacidad de generalización.

- **Modelo Sencillo:** Conservación de un modelo básico como referencia, lo cual fue útil para comparar mejoras.
- **Ampliación del Dataset:** Inclusión de más imágenes para mejorar el entrenamiento y obtener un modelo más robusto y preciso.

Aprendizajes adquiridos

A lo largo del desarrollo de esta práctica, he adquirido varios aprendizajes clave que son fundamentales para futuros proyectos en el ámbito de la inteligencia artificial aplicada a la agricultura:

- **Simplicidad y Necesidad:** Uno de los aprendizajes más importantes es que no hace falta el modelo más complejo para comenzar a observar resultados. Es crucial ajustarse a la necesidad específica del problema a resolver y no complicar en exceso la arquitectura del modelo desde el inicio.

- **Limitaciones Físicas y de Desarrollo:** Aunque la imaginación y la creatividad son esenciales para el desarrollo de soluciones innovadoras, siempre es necesario tener en cuenta las limitaciones físicas y de desarrollo. Esto incluye considerar el hardware disponible, los recursos computacionales y el tiempo necesario para entrenar y validar los modelos.
- **Respaldo y Guardado de Información:** Aprender a generar respaldos de manera correcta y guardar valores importantes de los modelos es fundamental para evitar la pérdida accidental de información difícil de conseguir. Esto garantiza que el trabajo realizado se pueda recuperar y continuar sin contratiempos.
- **Fuentes de Aprendizaje y Herramientas:** Aunque el campo de la IA sigue en crecimiento, existen numerosas fuentes y herramientas que facilitan el aprendizaje y el desarrollo de proyectos. Estas herramientas permiten comprender el trabajo de otras personas y aplicarlo a nuestros propios proyectos de manera eficiente.
- **Precaución con Google Colab:** A pesar de ser una herramienta útil, es importante no confiar completamente en Google Colab para proyectos críticos. Existen limitaciones y riesgos de pérdida de progreso, por lo que es recomendable considerar alternativas y tener siempre un plan de respaldo.
- **Selección de Experimentos:** Cuando la potencia de cómputo es una limitante, es crucial seleccionar cuidadosamente los experimentos a realizar. Esto ayuda a cumplir con los tiempos y formas establecidos, optimizando el uso de los recursos disponibles y asegurando resultados más eficientes.

Estos aprendizajes no solo han mejorado mi capacidad para desarrollar proyectos de IA, sino que también me han preparado mejor para enfrentar futuros desafíos en este campo. La iteración y el aprendizaje continuo son esenciales para el éxito en la inteligencia artificial y en cualquier otra área de la tecnología.

Contacto

Para cualquier duda favor de contactar:

marcosrafael2000@hotmail.com
+52 5537149673

Referencias

N. A. Mohammed, M. H. Abed, and A. T. Albu-Salih, Çonvolutional neural network for color images classification,"*Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1343-1349, Jun. 2022, doi: 10.11591/eei.v11i3.3730.

D. Qiu, T. Guo, S. Yu, W. Liu, L. Li, Z. Sun, H. Peng, and D. Hu, Çlassification of Apple Color and Deformity Using Machine Vision Combined with CNN,"*Special Issue: Multi- and Hyper-Spectral Imaging Technologies for Crop Monitoring*.

S. Lu, Z. Lu, S. Aok, and L. Graham, "Fruit Classification Based on Six Layer Convolutional Neural Network,"*School of Computer Science and Technology, Nanjing Normal University*, Nanjing, Jiangsu 210023, China; *School of Education Science, Nanjing Normal University*, Nanjing, Jiangsu 210023, China; *Faculty of Science and Technology, Angkor University*, Borey Seang Nam, Siem Reap Province, Cambodia; *Department of Mathematics, Ryerson University*, Toronto, ON M5B 2K3, Canada.

R. F. Rachmadi and I. K. E. Purnama, "Vehicle Color Recognition using Convolutional Neural Network,"*Department of Multimedia and Networking Engineering, Institut Teknologi Sepuluh Nopember*, Surabaya, Indonesia 60111. Email: fuad@its.ac.id, ketut@te.its.ac.id.