ENTREGA FINAL PROYECTO PDI

MIGUEL ANGEL LEON, ISABELA DURAN, SANTIAGO BERNAL

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
PROGRAMA PROCESAMIENTO DIGITAL DE IMÁGENES

PROFESOR: NATALI JOHANA VELANDIA

NOVIEMBRE 2023

1. Problemática

- Problemática social:

La problemática social que aborda este proyecto está estrechamente relacionada con la dificultad que enfrentan las personas con discapacidad visual en la autenticación y manejo de billetes en su vida diaria. La incapacidad para identificar con precisión los valores monetarios puede resultar en situaciones de vulnerabilidad financiera y limitar la autonomía de estas personas en la sociedad. Esta dificultad se vuelve aún más significativa en un entorno donde la independencia económica es crucial para la inclusión y la toma de decisiones personales.

El reconocimiento de billetes mediante una cámara web se presenta como una solución relevante para abordar esta problemática. Este sistema tiene como objetivo principal proporcionar a las personas con discapacidad visual una herramienta que les permita identificar y distinguir los billetes de forma rápida y precisa, ofreciéndoles independencia en la gestión de su dinero y promoviendo su inclusión social en el entorno financiero.

El desarrollo de un reconocedor de billetes colombianos no solo representa una solución tecnológica avanzada, sino también una contribución significativa para la igualdad de oportunidades y el acceso a la información económica, fortaleciendo así la independencia y calidad de vida de las personas con discapacidad visual en la sociedad colombiana.

- Problemática de aprendizaje automático:

El desarrollo de sistemas de reconocimiento de billetes mediante el uso de redes neuronales convolucionales plantea desafíos fundamentales y consideraciones especializadas. En el caso específico de la identificación de billetes colombianos de denominaciones como 2 mil, 5 mil y 10 mil pesos, surgen diversos retos que requieren una atención detallada y estrategias específicas para lograr una identificación precisa y confiable.

La complejidad de este problema radica en la variabilidad inherente a las imágenes de billetes, influenciada por factores como la iluminación, el ángulo de captura, el estado de los billetes (incluyendo desgaste, deterioro o variaciones en la calidad del papel) y la diversidad en las características visuales entre distintas denominaciones. Estos elementos añaden dificultad al proceso de detección y clasificación precisa de los billetes.

La naturaleza multiclase del problema, al intentar distinguir entre las diferentes denominaciones de billetes, presenta un reto adicional, ya que el modelo debe aprender a discernir entre características sutiles y, a veces, similares entre las clases. Asimismo, la disponibilidad limitada de datos etiquetados y el posible desbalance entre las clases pueden afectar la capacidad del modelo para generalizar adecuadamente a nuevas instancias de billetes.

La consideración del estado y la condición de los billetes como una variable adicional introduce una capa de complejidad, dado que el deterioro o la variación en la calidad del papel pueden afectar la apariencia y las características visuales, dificultando la tarea de identificación para el modelo.

Abordar eficazmente estos desafíos implica el diseño cuidadoso de estrategias de preprocesamiento de datos, selección de estructuras de redes neuronales adecuadas y la implementación de técnicas avanzadas de aprendizaje automático y procesamiento de imágenes. Estas consideraciones son esenciales para lograr un sistema de reconocimiento de billetes preciso, robusto y confiable."

2. Metodología utilizada para la recopilación y procesamiento de de datos

Recopilación de datos:

La obtención de datos para la construcción y entrenamiento del modelo de reconocimiento de billetes se realizó mediante una combinación de fuentes disponibles en internet y adquisición de imágenes en el entorno cotidiano. Dado que no se encontraron conjuntos de datos específicos que incluyeran todas las denominaciones necesarias, se llevó a cabo un proceso de recopilación exhaustivo.

La recopilación de datos en línea se realizó mediante la búsqueda y selección de imágenes de billetes colombianos disponibles en diferentes fuentes. Estas imágenes se utilizaron como un recurso inicial para la construcción del conjunto de datos de entrenamiento y validación.

Además, para enriquecer y diversificar el conjunto de datos, se implementó un enfoque práctico donde se capturaron imágenes de los billetes correspondientes en situaciones del día a día. Se tomaron fotografías de los billetes desde distintos ángulos y condiciones de iluminación, permitiendo así obtener una variedad de imágenes que reflejen la variabilidad real de los billetes en su uso cotidiano.

El proceso de captura de imágenes se llevó a cabo con el objetivo de abarcar diversas variaciones, incluyendo diferencias en el estado y la calidad de los billetes, así como variaciones en la iluminación y el ángulo de captura, con el fin de aumentar la robustez del modelo ante condiciones variables.

Esta metodología combinada de obtención de datos proporcionó un conjunto de imágenes diversificado y representativo, fundamental para el entrenamiento y evaluación del modelo de reconocimiento de billetes."

Preprocesamiento de datos:

- **Listado y Ordenamiento de Directorios:**

El proceso comienza con el uso de os.listdir para obtener una lista de elementos en el directorio especificado (/content/drive/MyDrive/PROYECTO/). Esta lista se ordena alfabéticamente para garantizar coherencia en la lectura de los subdirectorios.

- **Iteración sobre Subdirectorios y Archivos:**

A continuación, se inicia un bucle que itera sobre cada subdirectorio en la lista ordenada. Dentro de cada subdirectorio, se emplea glob.glob para obtener una lista de nombres de archivos que coinciden con el patrón '*.jpg'.

- **Procesamiento de Imágenes:**

Para cada archivo encontrado, se realiza el siguiente procesamiento:

Se extrae la etiqueta (etiqueta) del subdirectorio actual, considerando que cada subdirectorio representa una clase única.

La imagen se lee usando OpenCV (cv2.imread) y se convierte de BGR a RGB para asegurar el orden correcto de los canales.

Posteriormente, se redimensiona la imagen a un tamaño específico (IMG_SIZE = (224, 224)), comúnmente usado en tareas de clasificación de imágenes.

Los píxeles de la imagen se normalizan dividiendo cada valor por 255.0, escalando así los valores a un rango entre 0 y 1.

- **Almacenamiento de Datos en una Lista:**

Para cada imagen procesada, se agrega una tupla (imagen, etiqueta) a una lista llamada data.

- **Creación de un DataFrame:**

Finalmente, utilizando la biblioteca Pandas, se crea un DataFrame llamado carac con dos columnas: 'Características', que almacena las imágenes, y 'Etiqueta', que almacena las etiquetas correspondientes.

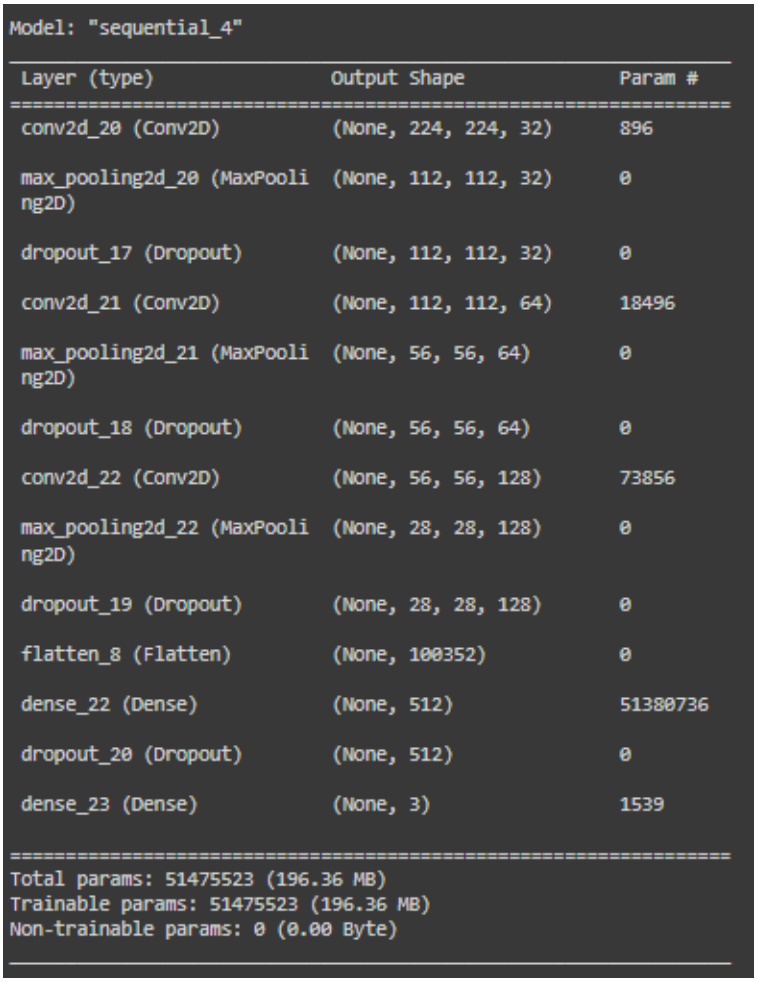
El uso de Pandas en este código permite organizar eficientemente los datos de las imágenes en una estructura tabular llamada DataFrame. Pandas simplifica la manipulación y preparación de datos, proporcionando funciones intuitivas para acceder, filtrar y transformar la información. Además, su compatibilidad con bibliotecas de aprendizaje automático, como Keras y scikit-learn

3. Detalles de las Estructuras de Modelos Implementados

- **Explicación Estructuras:**

1. **Primera estructura:**

La primera estructura es creada desde 0 por nosotros, a continuación se muestra una imagen sel summary:



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 224, 224, 32)      896

 max_pooling2d_20 (MaxPooli  (None, 112, 112, 32)      0
 ng2D)

 dropout_17 (Dropout)        (None, 112, 112, 32)      0

 conv2d_21 (Conv2D)          (None, 112, 112, 64)      18496

 max_pooling2d_21 (MaxPooli  (None, 56, 56, 64)        0
 ng2D)

 dropout_18 (Dropout)        (None, 56, 56, 64)        0

 conv2d_22 (Conv2D)          (None, 56, 56, 128)       73856

 max_pooling2d_22 (MaxPooli  (None, 28, 28, 128)       0
 ng2D)

 dropout_19 (Dropout)        (None, 28, 28, 128)       0

 flatten_8 (Flatten)         (None, 100352)            0

 dense_22 (Dense)            (None, 512)               51380736

 dropout_20 (Dropout)        (None, 512)               0

 dense_23 (Dense)            (None, 3)                 1539

=================================================================
Total params: 51475523 (196.36 MB)
Trainable params: 51475523 (196.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

La estructura del modelo propuesto presenta varias características clave que contribuyen a su eficacia en tareas de clasificación de imágenes. Aquí se detallan algunos de los puntos más relevantes:

Capas Convolutivas y MaxPooling:
La presencia de capas convolutivas (Conv2D) permite al modelo aprender patrones locales en las imágenes, extrayendo características importantes. El MaxPooling reduce la dimensionalidad, conservando las características más importantes y mejorando la eficiencia computacional.

Función de Activación ReLU:
La elección de la función de activación ReLU (Rectified Linear Unit) en las capas convolutivas introduce no linealidades en el modelo. ReLU ha demostrado ser efectiva al superar el problema de desvanecimiento del gradiente y acelerar el entrenamiento.

Regularización L2:
La regularización L2 (también conocida como weight decay) se aplica para evitar el sobreajuste del modelo. Al agregar un término de penalización a los pesos, se desincentiva el crecimiento excesivo de los mismos durante el entrenamiento, lo que ayuda a generalizar mejor a nuevos datos.

Capa de Dropout:
La inclusión de capas de Dropout después de las capas convolutivas y completamente conectadas contribuye a la regularización adicional. Dropout apaga aleatoriamente neuronas durante el entrenamiento, previniendo la dependencia excesiva entre neuronas y mejorando la robustez del modelo.

Capa Flatten y Capas Densas:
La capa Flatten transforma la salida de las capas convolutivas en un vector unidimensional antes de conectarlo a capas densas. Las capas densas aprenden patrones globales y relaciones más complejas en los datos.

Función de Activación Softmax:
La función de activación Softmax en la capa de salida es ideal para problemas de clasificación múltiple. Produce una distribución de probabilidad sobre las clases, facilitando la interpretación de las salidas del modelo como probabilidades.

El uso de padding 'Same' asegura que la salida de la capa convolutiva tenga la misma dimensión espacial que la entrada. Esto es útil cuando se busca conservar la información en los bordes de las imágenes, evitando la pérdida de información relevante durante las convoluciones.

En conclusión la combinación de capas convolutivas, funciones de activación apropiadas, regularización y Dropout hace que el modelo sea robusto, capaz de aprender representaciones significativas y generalizar bien a nuevos datos. La elección de estas técnicas se basa en principios ampliamente respaldados en la literatura de aprendizaje profundo, proporcionando un equilibrio efectivo entre complejidad y capacidad predictiva.

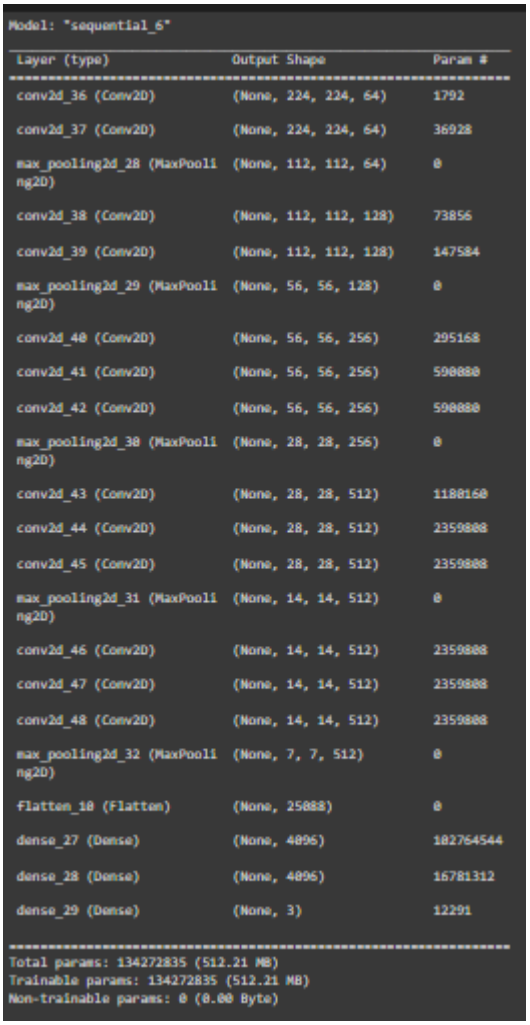
Data Augmentation:
En esta parte, se utilizan técnicas de aumento de datos para mejorar la capacidad del modelo para reconocer diferentes variaciones de imágenes. Esto incluye rotar las imágenes hasta 20 grados, desplazarlas horizontal y verticalmente, y voltearlas horizontalmente (lo que sería como ver la imagen en un espejo). Esta variedad de cambios hace que el modelo sea más robusto y capaz de reconocer mejor los billetes en diversas condiciones.

Hiperparámetros:

- Learning Rate (Tasa de Aprendizaje): Define cuán rápido o lento el modelo ajusta sus parámetros durante el entrenamiento. Una tasa alta puede acelerar el aprendizaje, pero si es demasiado alta, puede hacer que el modelo se salte la solución óptima, en este caso usamos 0.001 determinamos que nos daba el mejor resultado a prueba y error.
- Batch Size (Tamaño del Lote): Determina cuántas imágenes se utilizan en cada paso de entrenamiento. Un tamaño de lote más grande puede acelerar el entrenamiento, pero puede requerir más memoria en este caso usamos 32, determinamos que nos daba el mejor resultado a prueba y error y también nos basamosl en algunas páginas que consultamos y estarán abajo en las citas.
- Epochs (Épocas): Indica cuántas veces el modelo ve el conjunto completo de datos durante el entrenamiento. Más épocas pueden ayudar al modelo a aprender más detalles de los datos, en este caso usamos 50, fue los mas optimo aunque al tener EarlyStopping no nos complicamos tanto con las épocas.

Entrenamiento del Modelo:
Aquí es donde realmente se "enseña" al modelo. Se utiliza un concepto llamado EarlyStopping para evitar que el modelo se sobreajuste (es decir, que aprenda demasiado bien solo los datos de entrenamiento y no generalice bien a datos nuevos). Se compila el modelo para definir cómo se evaluará y optimizará. Luego, se entrena el modelo utilizando los datos de entrenamiento y validación, observando su progreso a través de diferentes épocas. Al final del entrenamiento, se evalúa su rendimiento en los datos de validación para ver qué tan bien aprendió, en total este modelo se demoró alrededor de 8 minutos obteniéndose resultados muy buenos en la primera que lo entrenamos, este es uno de los modelos que usamos para predecir en página web y la verdad funciono bien, hay cosas por mejorar aunque esto va mas relacionado al dataset, también se podría mejorar un poco más con algunas capas convolucionales más y algunos maxpooling con el fin de que varie un poco más los datos sin caer en que se vuelva una estructura muy compleja, en conclusión es una buena red, nos funcionó, pero podría ser mucho mejor a continuación se muestran algunos datos.
(cabe resaltar que las siguientes imágenes no son del primer modelo que tuvimos y el cual nos dio buen resultado, las siguientes imágenes son una de las tantas veces que entrenamos el código).

2. **Segunda estructura:**



```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_36 (Conv2D)          (None, 224, 224, 64)      1792

 conv2d_37 (Conv2D)          (None, 224, 224, 64)      36928

 max_pooling2d_28 (MaxPooli  (None, 112, 112, 64)      0
 ng2D)

 conv2d_38 (Conv2D)          (None, 112, 112, 128)     73856

 conv2d_39 (Conv2D)          (None, 112, 112, 128)     147584

 max_pooling2d_29 (MaxPooli  (None, 56, 56, 128)       0
 ng2D)

 conv2d_40 (Conv2D)          (None, 56, 56, 256)       295168

 conv2d_41 (Conv2D)          (None, 56, 56, 256)       590080

 conv2d_42 (Conv2D)          (None, 56, 56, 256)       590080

 max_pooling2d_30 (MaxPooli  (None, 28, 28, 256)       0
 ng2D)

 conv2d_43 (Conv2D)          (None, 28, 28, 512)       1180160

 conv2d_44 (Conv2D)          (None, 28, 28, 512)       2359808

 conv2d_45 (Conv2D)          (None, 28, 28, 512)       2359808

 max_pooling2d_31 (MaxPooli  (None, 14, 14, 512)       0
 ng2D)

 conv2d_46 (Conv2D)          (None, 14, 14, 512)       2359808

 conv2d_47 (Conv2D)          (None, 14, 14, 512)       2359808

 conv2d_48 (Conv2D)          (None, 14, 14, 512)       2359808

 max_pooling2d_32 (MaxPooli  (None, 7, 7, 512)         0
 ng2D)

 flatten_10 (Flatten)        (None, 25088)             0

 dense_27 (Dense)            (None, 4096)              102764544

 dense_28 (Dense)            (None, 4096)              16781312

 dense_29 (Dense)            (None, 3)                 12291

=================================================================
Total params: 134272835 (512.21 MB)
Trainable params: 134272835 (512.21 MB)
Non-trainable params: 0 (0.00 Byte)
```

La elección de utilizar la estructura VGG16 en lugar de la anterior se basa en la complejidad y profundidad de la red. VGG16 es conocida por su profundidad y capacidad para aprender patrones jerárquicos complejos en datos visuales. Sin embargo, aunque VGG16 ha demostrado ser efectiva en grandes conjuntos de datos como ImageNet, su rendimiento puede verse limitado cuando se trabaja con conjuntos de datos más pequeños.

Aquí están algunas razones para explicar la elección y las modificaciones realizadas:

Complejidad del Modelo y Tamaño del Conjunto de Datos:

VGG16 es una red neuronal profunda con múltiples capas, diseñada originalmente para grandes conjuntos de datos como ImageNet.
Para conjuntos de datos más pequeños, el uso de una red más profunda como VGG16 podría llevar a sobreajuste debido a la complejidad del modelo en relación con la cantidad limitada de datos disponibles.
Problema de Sobreajuste:

Reducir la complejidad del modelo puede ayudar a prevenir el sobreajuste en conjuntos de datos pequeños. Algunas capas y conexiones se eliminaron para simplificar el modelo y evitar un ajuste excesivo.
Tiempo de Entrenamiento:

Entrenar modelos más grandes, como VGG16, puede requerir más tiempo y recursos computacionales. La elección de un modelo más simple puede hacer que el proceso de entrenamiento sea más eficiente.
Adición de Capas y Regularización:

Se agregaron capas de regularización (Dropout y L2) para ayudar a controlar el sobreajuste y mejorar la generalización del modelo.
Estas capas proporcionan herramientas para reducir la complejidad del modelo y prevenir el sobreajuste.
Adaptación a Datos Específicos:

La estructura se modificó para adaptarse mejor a las características específicas del conjunto de datos en cuestión.

En conclusión descartamos vgg16 y decidimos usar una estructura dada por la profesora Natali Johana Velandia Fajardo en un código cnn dado por ella en clase, lo acoplamos hicimos data augmentation y nos fue mucho mejor, a continuación se muestra la estructura usada:

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 224, 224, 3)]     0

 conv2d_49 (Conv2D)          (None, 220, 220, 6)       456

 max_pooling2d_33 (MaxPooli  (None, 110, 110, 6)       0
 ng2D)

 conv2d_50 (Conv2D)          (None, 106, 106, 16)      2416

 max_pooling2d_34 (MaxPooli  (None, 53, 53, 16)        0
 ng2D)

 flatten_11 (Flatten)        (None, 44944)             0

 dense_30 (Dense)            (None, 120)               5393400

 dense_31 (Dense)            (None, 84)                10164

 dense_32 (Dense)            (None, 3)                 255

=================================================================
Total params: 5406691 (20.62 MB)
Trainable params: 5406691 (20.62 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Este modelo de red neuronal convolucional (CNN) se basa en la estructura LeNet-5, un diseño clásico desarrollado por Yann LeCun para tareas de reconocimiento de patrones. Aquí se presenta un análisis argumentativo de las elecciones de diseño y parámetros específicos:

Capas Convolucionales y Pooling:
Se utilizan dos capas convolucionales seguidas de capas de max pooling. Estas capas permiten extraer características importantes de las imágenes, proporcionando robustez frente a las variaciones en la posición de los objetos.

Funciones de Activación:
La función de activación 'ReLU' (Rectified Linear Unit) se elige para introducir no linealidades en el modelo. ReLU es eficiente en términos computacionales y ha demostrado ser efectiva en la práctica al permitir que la red aprenda de manera más rápida y efectiva.

Regularización L2:
La regularización L2 se aplica para mitigar el riesgo de sobreajuste al penalizar los valores extremos de los pesos. Esta técnica ayuda a mejorar la generalización del modelo y su capacidad para trabajar con datos no vistos.

Capas Densas:

Dos capas densas (fully connected) se utilizan para la clasificación final. Estas capas se encargan de combinar las características extraídas por las capas convolucionales y aprenden a mapear las representaciones intermedias a las clases de salida.

Softmax para Clasificación Multiclase:
La función de activación 'Softmax' en la capa de salida se utiliza para clasificación multiclase. Proporciona probabilidades normalizadas para cada clase, facilitando la interpretación y la toma de decisiones.

Elección de Tamaño de Filtros y Capas Densas:
Los tamaños de los filtros (5x5) y las dimensiones de las capas densas (120 y 84) se seleccionan basándose en consideraciones heurísticas y experiencias anteriores. Estos valores son comunes en estructuras clásicas y han demostrado ser efectivos en tareas de clasificación de imágenes.

En resumen, este modelo se seleccionó por su simplicidad y eficacia en tareas de clasificación de imágenes. La combinación de capas convolucionales para la extracción de características y capas densas para la clasificación proporciona un equilibrio adecuado para el problema en cuestión. La regularización L2 y las funciones de activación elegidas contribuyen a la estabilidad del modelo y su capacidad para generalizar bien a nuevos datos.

Data Augmentation:
En esta sección, se lleva a cabo el aumento de datos utilizando la función ImageDataGenerator de Keras. Se configuran varias transformaciones para aumentar la variedad y cantidad de imágenes disponibles para entrenar el modelo. Se aplican rotaciones de hasta 20 grados, desplazamientos horizontales y verticales de hasta el 20% del tamaño de la imagen y se activa el volteo horizontal. Estos cambios se aplican a las imágenes de entrenamiento, lo que ayuda al modelo a aprender de una mayor diversidad de perspectivas y variaciones en los billetes.

EarlyStopping y Compilación del Modelo:
Para evitar el sobreajuste del modelo (cuando este se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos), se utiliza la técnica de EarlyStopping. Esta técnica monitorea la pérdida en los datos de validación y detiene el entrenamiento si la pérdida no mejora después de un número específico de épocas (en este caso, se detiene después de 5 épocas sin mejora). Además, se compila el modelo utilizando la función de pérdida categorical_crossentropy, el optimizador Adam y se define la métrica de precisión para evaluar su desempeño.

Entrenamiento y Evaluación del Modelo:
A continuación, se entrena el modelo utilizando los datos de entrenamiento y validación a través del método fit. El modelo observa las imágenes aumentadas y, después de cada época, evalúa su rendimiento en los datos de validación para controlar el sobreajuste. Se muestra el

tiempo total de entrenamiento y al final se evalúa la precisión y pérdida finales en los datos de validación.

Tercera estructura:

```
# Convertir etiquetas a formato one-hot
y_train_one_hot = to_categorical(y_train, num_classes=3)
y_test_one_hot = to_categorical(y_test, num_classes=3)

# Definir la arquitectura del modelo MobileNetV2
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Congelar las capas convolucionales del modelo base
for layer in base_model.layers:
    layer.trainable = False

# Agregar capas personalizadas para la clasificación
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(256, activation='relu')(x)
x = Dropout(0.35)(x)
output = Dense(3, activation='softmax')(x)  # 3 clases de billetes

# Crear el modelo completo
model3 = Model(inputs=base_model.input, outputs=output)
```

El modelo implementa la estructura MobileNetV2 preentrenada en el conjunto de datos 'imagenet' y se personaliza para adaptarse a la tarea específica de clasificación de billetes. Aquí se presentan los puntos clave y la justificación de las decisiones tomadas:

Transferencia de Aprendizaje:

Se opta por MobileNetV2 como base del modelo debido a su eficiencia y rendimiento en tareas de clasificación de imágenes. Esta estructura está diseñada para dispositivos móviles, lo que la hace liviana y adecuada para entrenamientos en recursos limitados.
Congelación de Capas Convolucionales:

Todas las capas convolucionales de MobileNetV2 se congelan para retener los conocimientos previos adquiridos en el conjunto de datos 'imagenet'. Esta decisión se toma para aprovechar las características generales aprendidas por la red preentrenada.
Capas Personalizadas:

Se añaden capas personalizadas para adaptar el modelo a la tarea específica de clasificación de billetes. La capa GlobalAveragePooling2D se utiliza para reducir las dimensiones de la representación espacial. A esto le sigue una capa densa con 256 neuronas y función de activación 'ReLU', seguida de una capa de Dropout para mitigar el sobreajuste.

Softmax para Clasificación Multiclase:

La capa de salida utiliza la función de activación 'Softmax' para proporcionar probabilidades normalizadas para las tres clases de billetes.
Optimizador Adam y Función de Pérdida Categórica Cruzada:

El modelo se compila con el optimizador Adam y la función de pérdida 'categorical_crossentropy' para la clasificación multiclase. Adam se elige por su eficiencia y adaptabilidad a diferentes tipos de datos.
Regularización y Dropout:

La regularización L2 no se utiliza directamente en este modelo debido a la congelación de las capas convolucionales pre entrenadas. Sin embargo, se incorpora Dropout después de la capa densa para evitar el sobreajuste durante el entrenamiento.
EarlyStopping:

Se implementa la técnica EarlyStopping con un monitor en la pérdida de validación ('val_loss'), paciencia de 5 epochs y restauración de los mejores pesos. Esto ayuda a evitar el sobreajuste y a detener el entrenamiento cuando no se observan mejoras.
Valores Específicos en el Fit y EarlyStopping:

Se eligen valores específicos en el entrenamiento, como 20 epochs, una tasa de aprendizaje de 0.0001, y los parámetros del EarlyStopping mencionados anteriormente, para equilibrar la eficiencia y la capacidad de generalización del modelo.
En conclusión, este modelo MobileNetV2 personalizado se fundamenta en la transferencia de aprendizaje, aprovechando la capacidad de la estructura preentrenada para extraer características relevantes. La adaptación a la tarea específica se logra mediante capas personalizadas y técnicas de regularización. El uso de Adam como optimizador y la función de pérdida 'categorical_crossentropy' son elecciones comunes en tareas de clasificación. La implementación de EarlyStopping contribuye a un entrenamiento más eficiente y evita el sobreajuste, permitiendo obtener un modelo robusto con un rendimiento óptimo.

4. Selección de Estructura para Despliegue y Diagrama de Metodología Completa

Estructura para despliegue en Flask:

La estructura de despliegue se desprende principalmente en dos partes:

1. **Backend (app.py):**

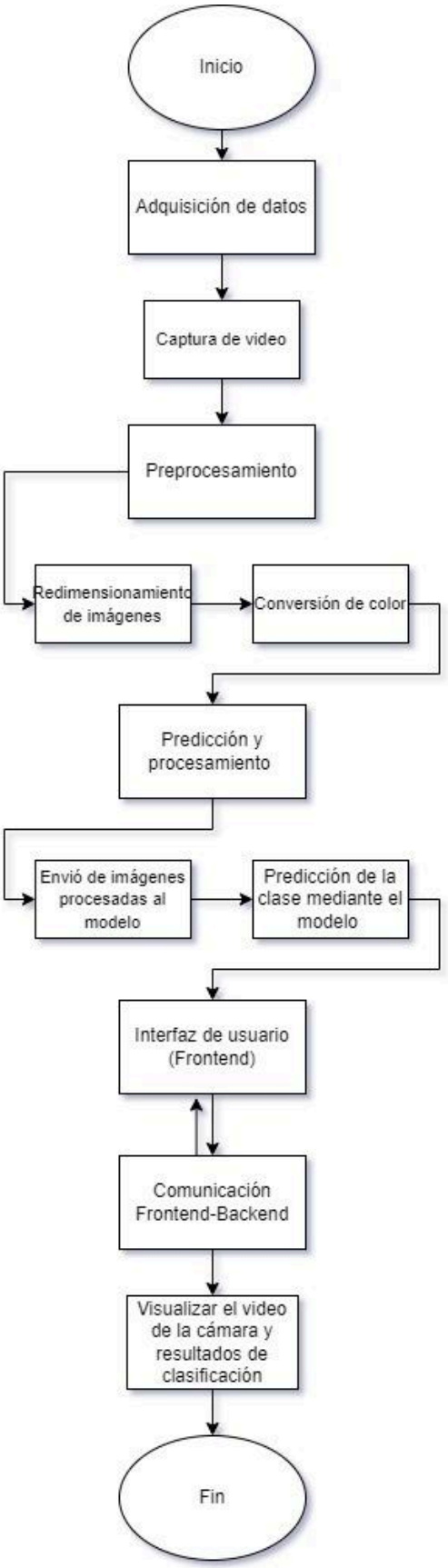
- Utiliza Flask, un micro framework de Python, para manejar las rutas y solicitudes HTTP. Flask proporciona una estructura para construir aplicaciones web y manejar las interacciones del servidor.

- Carga un modelo de aprendizaje automático previamente entrenado para la clasificación de billetes. Se utiliza la librería Keras para cargar el modelo (pre entrenado en la detección de billetes) y realizar predicciones con imágenes.
- Define dos rutas principales:
  '/': es la ruta principal que renderiza el archivo HTML 'index.html'. Esta ruta es la página de inicio que muestra la interfaz al usuario. Al acceder a esta ruta, se carga la interfaz gráfica de la aplicación.
  '/classify_frame': es una ruta de tipo POST utilizada para recibir y procesar las imágenes capturadas desde la cámara. Cuando se recibe una imagen desde el frontend, se realiza el procesamiento de la imagen y se realiza la predicción del billete.

2. **Frontend (index.html):**

- Contiene el código HTML y JavaScript que construye la interfaz de usuario. Utiliza la API de medios (navigator.mediaDevices.getUserMedia) para acceder a la cámara del dispositivo. Esto permite obtener acceso a la cámara del usuario para capturar el video en tiempo real.
- Crea un elemento de vídeo que muestra el flujo de video de la cámara en la interfaz. Este elemento se actualiza constantemente para mostrar el video en tiempo real desde la cámara.
- Utiliza JavaScript para capturar los fotogramas del video y enviarlos al servidor para la clasificación utilizando fetch y FormData. Con intervalos regulares, se capturan los frames del video y se envían al servidor para ser procesados por el modelo de clasificación de billetes.
- Muestra dos secciones en la interfaz:
  'Billete detectado': sección que se actualiza con la predicción recibida del servidor. Esta área de la interfaz muestra el resultado de la clasificación del billete, que es enviada por el backend después de procesar la imagen.
  'Billete detectado en tiempo real': sección que muestra la predicción en tiempo real del billete detectado. Esta área se actualiza dinámicamente a medida que se reciben y procesan las predicciones del servidor, proporcionando una visualización en tiempo real de la detección de billetes.

Diagrama del flujo y metodología.


Inicio
Adquisición de datos
Captura de video
Preprocesamiento
Redimensionamiento de imágenes
Conversión de color
Predicción y procesamiento
Envió de imágenes procesadas al modelo
Predicción de la clase mediante el modelo
Interfaz de usuario (Frontend)
Comunicación Frontend-Backend
Visualizar el video de la cámara y resultados de clasificación
Fin

5. Descripción de la aplicación con Flask y su funcionamiento.

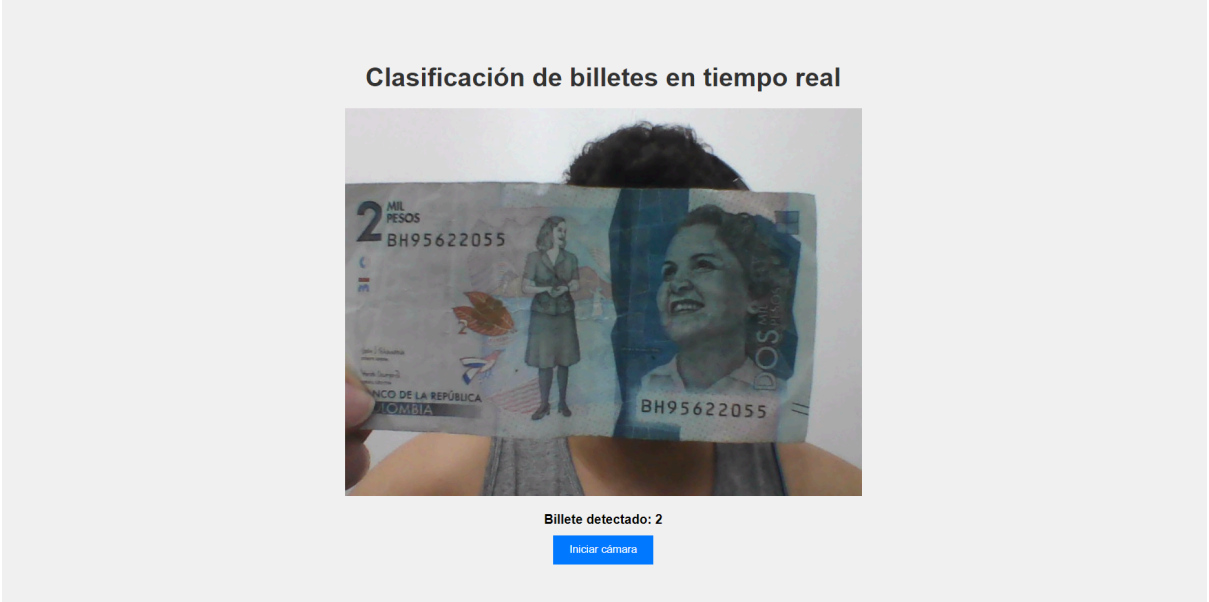
La aplicación desarrollada con Flask tiene como objetivo facilitar el reconocimiento de billetes a través de la cámara de un dispositivo. Utiliza un modelo de aprendizaje automático pre-entrenado para identificar y clasificar diferentes denominaciones de billetes, proporcionando una interfaz web para los usuarios.

- **Funcionamiento de la Aplicación:**

La aplicación web cuenta con una interfaz de usuario intuitiva que permite a los usuarios interactuar con ella de manera sencilla:

1. **Interfaz de Usuario:**

- Al acceder a la aplicación, los usuarios se encuentran con una interfaz web que muestra un video en tiempo real de la cámara.
- Una sección de la interfaz proporciona la predicción del billete detectado.



2. **Proceso de Reconocimiento:**

- Una vez iniciada la cámara, la aplicación captura fotogramas continuos de video que son enviados al backend (desarrollado con Flask) para su procesamiento.

- En el backend, los fotogramas se someten a un preprocesamiento para adaptarlos al formato requerido por el modelo de aprendizaje automático. Este proceso incluye la redimensión de las imágenes a un tamaño específico, la

normalización de los píxeles y la conversión al formato de imagen adecuado para el modelo.

- Posteriormente, se aplica el modelo de aprendizaje automático previamente entrenado sobre los fotogramas procesados para realizar la predicción de la denominación del billete detectado. Esta predicción se realiza en tiempo real

- Los resultados de la predicción, es decir, la denominación del billete detectado, se envían de vuelta al frontend de la aplicación para ser mostrados a los usuarios.

```python
app = Flask(__name__)
classifier = load_model('MODELO_CNN/modelitof.h5')
class_labels = ['10', '2', '5']
cap = cv2.VideoCapture(0)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/classify_frame', methods=['POST'])
def classify_frame():
    file = request.files['frame']
    img = cv2.imdecode(np.fromstring(file.read(), np.uint8), cv2.IMREAD_COLOR)
    img = cv2.resize(img, (224, 224))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Cambio de BGR a RGB
    img = img.astype('float') / 255.0
    img = np.reshape(img, (1, 224, 224, 3))

    preds = classifier.predict(img)[0]
    label = class_labels[preds.argmax()]

    return jsonify({'bill': label})
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Clasificación de billetes en tiempo real</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <h1>Clasificación de billetes en tiempo real</h1>
    <video id="video" width="640" height="480" autoplay></video> <!-- Mostrar el video -->
    <div id="prediction-result">Billete detectado:</div> <!-- Mostrar la predicción del billete aquí -->
    <button id="start-button">Iniciar cámara</button> <!-- Botón para iniciar la cámara -->

    <script>
        var video = document.getElementById('video');
        var predictionDiv = document.getElementById('prediction-result');
        var startButton = document.getElementById('start-button');
```

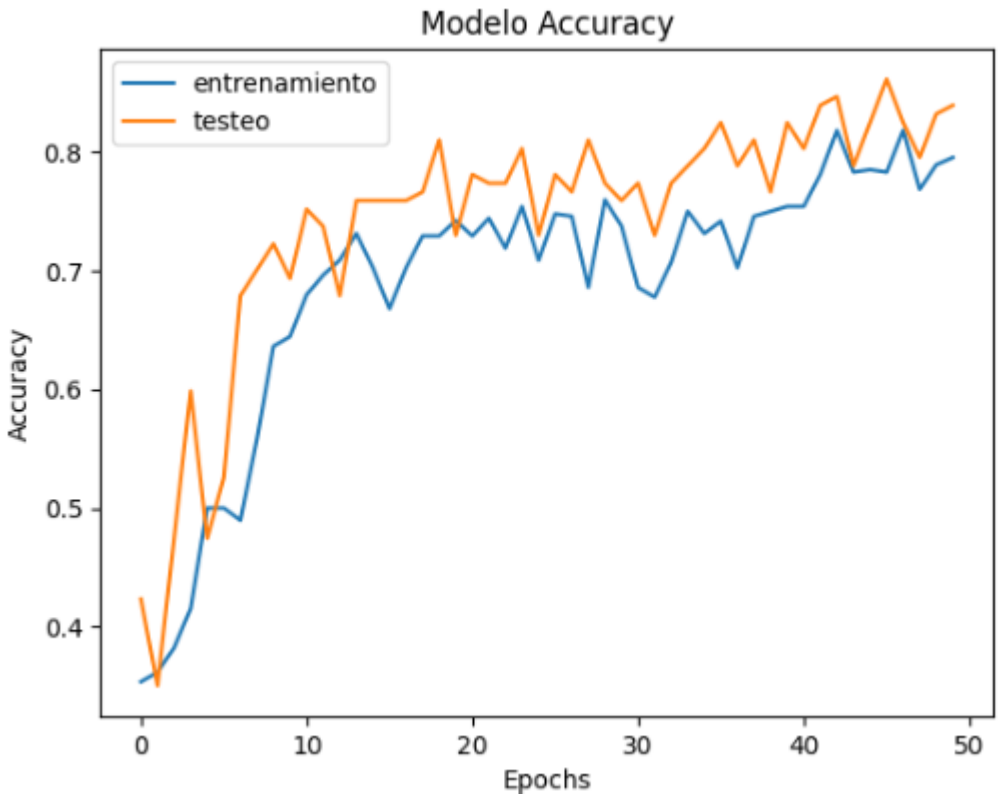
```
startButton.onclick = function() {
    // Solicitar el acceso a la cámara
    if (navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {
        navigator.mediaDevices.getUserMedia({ video: true }).then(function(stream) {
            video.srcObject = stream;
            setInterval(sendFrameToServer, 1000); // Enviar un fotograma cada segundo
        });
    }
};

function sendFrameToServer() {
    var canvas = document.createElement('canvas');
    canvas.width = 640;
    canvas.height = 480;
    var ctx = canvas.getContext('2d');
    ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
    canvas.toBlob(function(blob) {
        var formData = new FormData();
        formData.append('frame', blob);
        fetch('/classify_frame', {
            method: 'POST',
            body: formData,
        })
```

```
            })
            .then(response => response.json())
            .then(data => {
                predictionDiv.innerText = 'Billete detectado: ' + data.bill;
            });
        }, 'image/jpeg');
    }
</script>
</body>
</html>
```

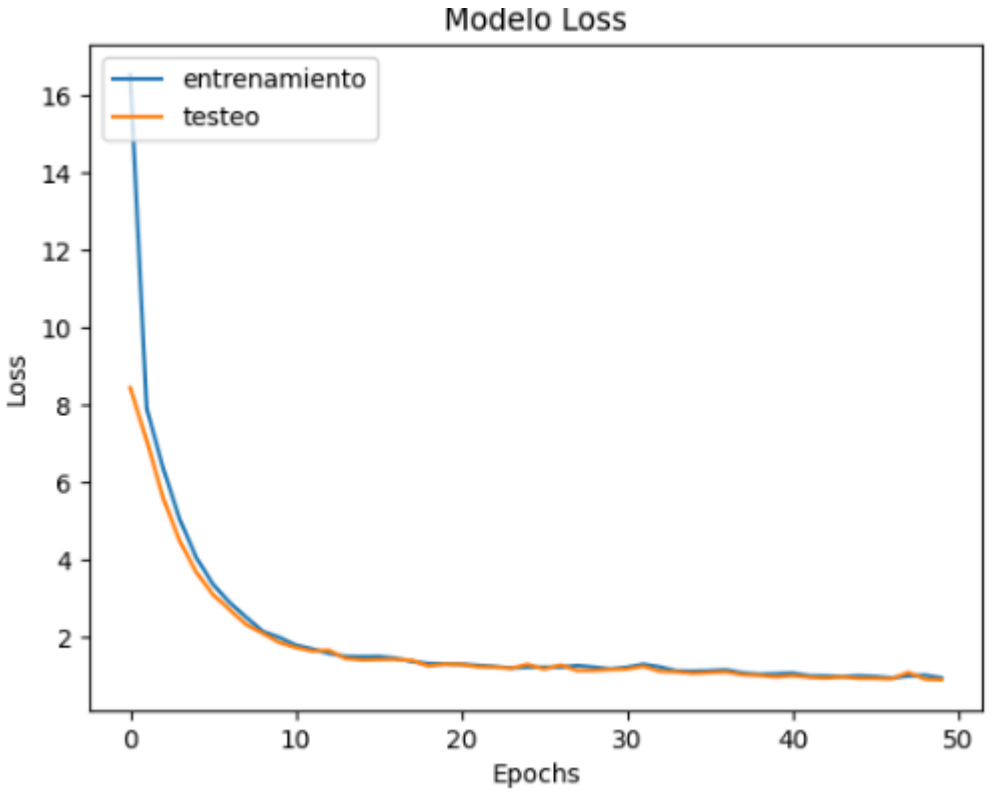
6. Resultados Experimentales y Análisis de Rendimiento.

Presentación de los resultados obtenidos al evaluar el modelo implementado.

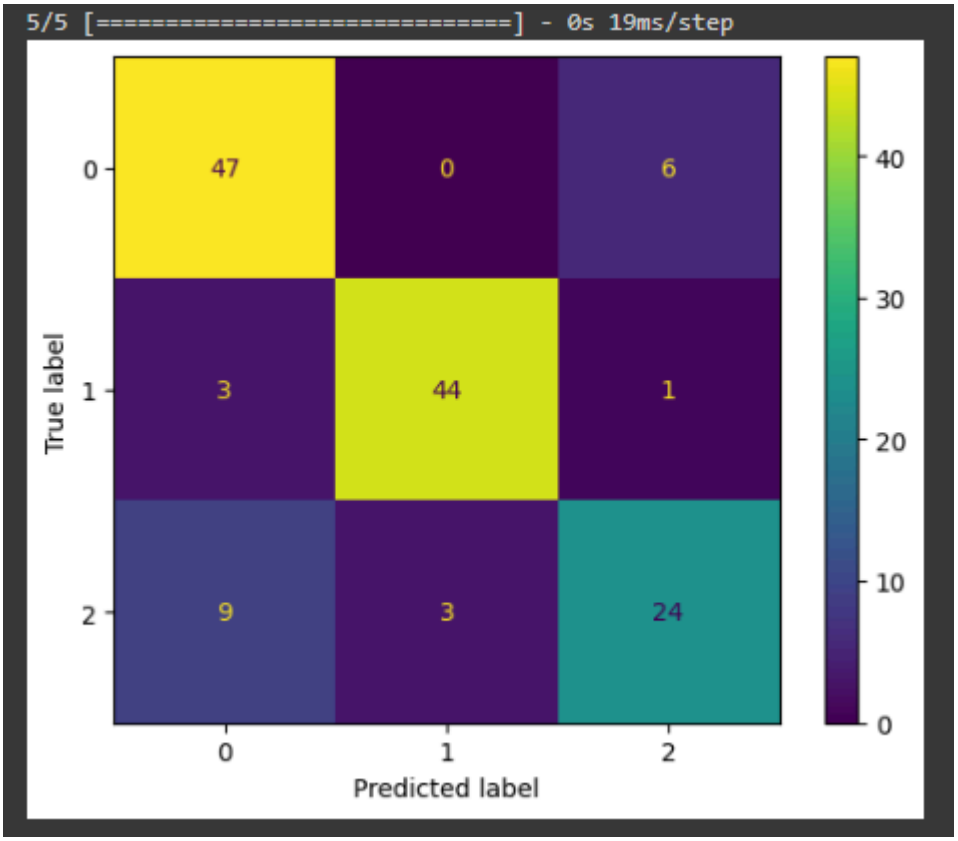
Primera estructura:



Esta gráfica del accuracy con muchos picos puede ser debido a varias cosas por ejemplo Overfitting, Learning Rate, Batch Size o Datos Ruidosos o Variabilidad en los Datos, probamos muchas cosas pero al final nos dimos cuenta que eran varias cosas juntas, por ejemplo en ese momento no teníamos padding, teníamos una Batch Size de 16 y no teníamos bien puesto el data augmentation, al final logramos corregir un poco pero sin mucho cambio, de igual forma el modelo siguió prediciendo bien y es uno de los buenos modelos que guardamos.



En este caso la gráfica de loss nos estaba dando bien sin mucho más que agregar o cambiar.



Aquí lo que podemos ver es que la etiqueta 2 que en nuestro caso es el billete de 5 mil se estaba confundiendo un poco con la etiqueta 0 que es el billete de 10, esto más que todo a que

en ese momento no teníamos padding = same y cuando lo agregamos tuvimos una mejora significativa, debido a que aumentaron las características.

```
5/5 [==============================] - 0s 19ms/step
              precision    recall  f1-score   support

          10       0.80      0.89      0.84        53
           2       0.94      0.92      0.93        48
           5       0.77      0.67      0.72        36

    accuracy                           0.84       137
   macro avg       0.84      0.82      0.83       137
weighted avg       0.84      0.84      0.84       137
```

Para la clase '10', la precisión es del 80%, lo que indica que el 80% de las instancias clasificadas como '10' son realmente '10'.
Para la clase '2', la precisión es del 94%, lo que significa que el 94% de las instancias clasificadas como '2' son realmente '2'.
Para la clase '5', la precisión es del 77%, indicando que el 77% de las instancias clasificadas como '5' son realmente '5'.

Para la clase '10', el recall es del 89%, lo que significa que el 89% de todas las instancias que son realmente '10' fueron correctamente identificadas.
Para la clase '2', el recall es del 92%, indicando que el 92% de las instancias que son realmente '2' fueron correctamente identificadas.
Para la clase '5', el recall es del 67%, indicando que el 67% de las instancias que son realmente '5' fueron correctamente identificadas.

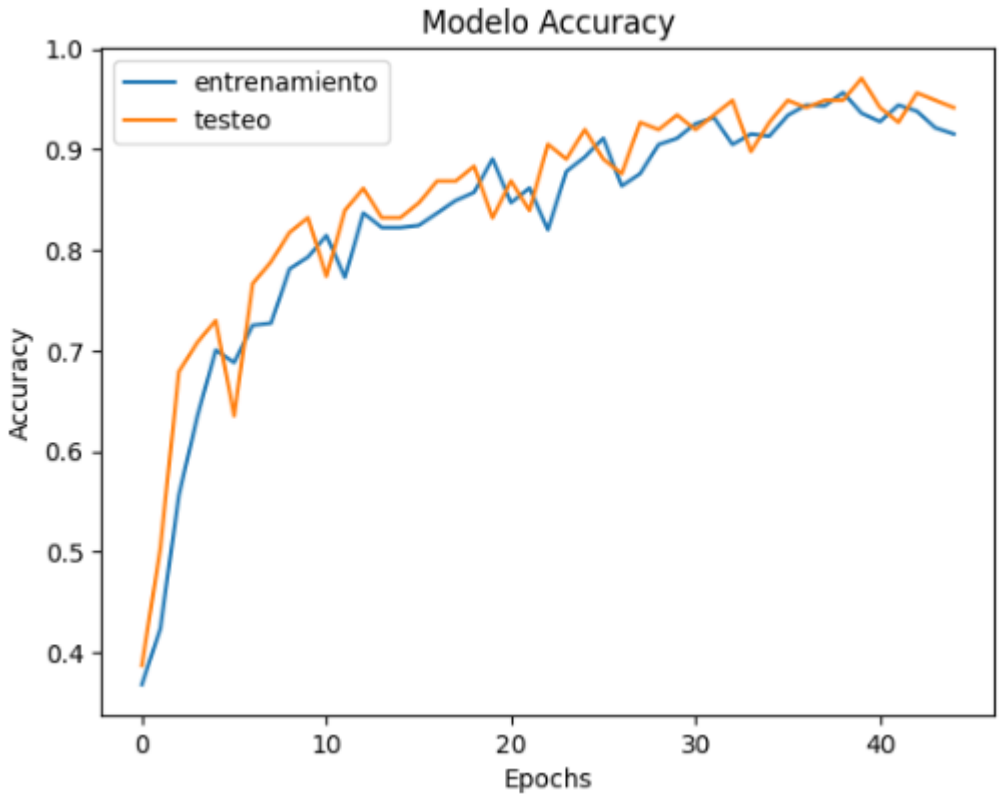
F1-score es la media armónica entre precisión y recall. Proporciona un equilibrio entre ambas métricas. En tu evaluación, los F1-scores son:
Para la clase '10', el F1-score es 0.84.
Para la clase '2', el F1-score es 0.93.
Para la clase '5', el F1-score es 0.72.
Cabe resaltar que es un modelo bueno con algunos buenos resultados, pero aun podría mejorar mucho más.

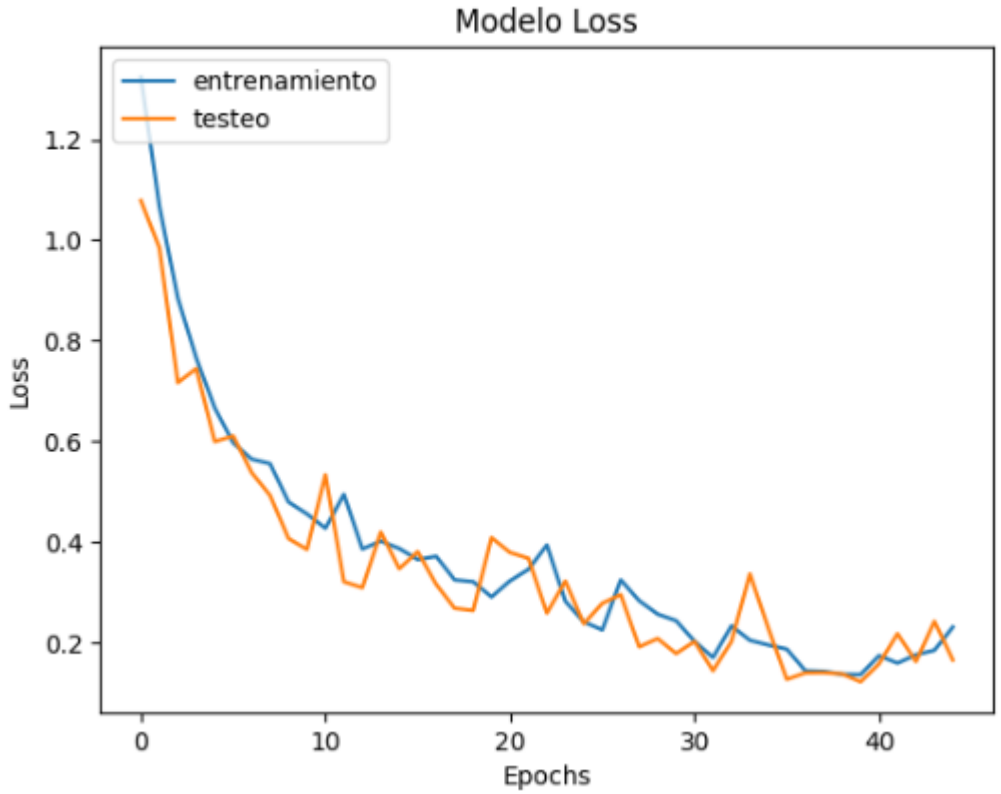
```python
# Ruta del archivo de imagen de prueba
img_path = '/content/twolukas.jpg'

# Leer la imagen y realizar preprocesamiento
img = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (224, 224))
img = img / 255.
img = np.reshape(img, [1, 224, 224, 3])

# Medir el tiempo de predicción
start_time = time.time()

# Hacer la predicción
resultado = model.predict(img)

# Calcular el tiempo transcurrido
elapsed_time = time.time() - start_time

# Obtener la etiqueta correspondiente a la predicción y la probabilidad asociada
res_word = etiquetas[(np.argmax(resultado, axis=1))[0]]
probabilidad = resultado[0][np.argmax(resultado, axis=1)] * 100

# Imprimir los resultados y el tiempo de predicción
print("Para el archivo:{0}, usted dijo:{1}, con un {2}% de certeza".format(img_path, res_word, probabilidad))
print("Tiempo de predicción: {:.4f} segundos".format(elapsed_time))
```

```
1/1 [==============================] - 0s 26ms/step
Para el archivo:/content/twolukas.jpg, usted dijo:1, con un [43.526974]% de certeza
Tiempo de predicción: 0.0868 segundos
```

Aquí tenemos una de las predicciones la cual está correcta, pero lo hace con un 43% de certeza.

Segunda estructura:





Teniendo en cuenta las dos gráficas anteriores, de loss y accuracy concluimos los siguientes puntos:

Puntos Positivos:

La estructura tiene una complejidad adecuada para el problema dado, con capas convolucionales que pueden extraer características jerárquicas de las imágenes y capas totalmente conectadas para la clasificación final.

El uso de aumentación de datos a través de ImageDataGenerator puede haber ayudado al modelo a generalizar mejor y mejorar su capacidad para manejar la variabilidad en los datos de entrenamiento.

La implementación de Early Stopping es una práctica positiva para evitar el sobreajuste y conservar el mejor modelo en función de la pérdida en el conjunto de validación.
Tiempo de Entrenamiento:

El tiempo total de entrenamiento es relativamente corto, lo que sugiere una convergencia rápida del modelo.

gracias a eso obtuvimos muy buenos resultados la verdad, a pesar de sus gráficas:

```
Tiempo total de entrenamiento: 389.37 segundos
5/5 [==============================] - 0s 14ms/step - loss: 0.1209 - accuracy: 0.9708
Pérdida final en datos de validación: 0.1209
Precisión final en datos de validación: 0.9708
```

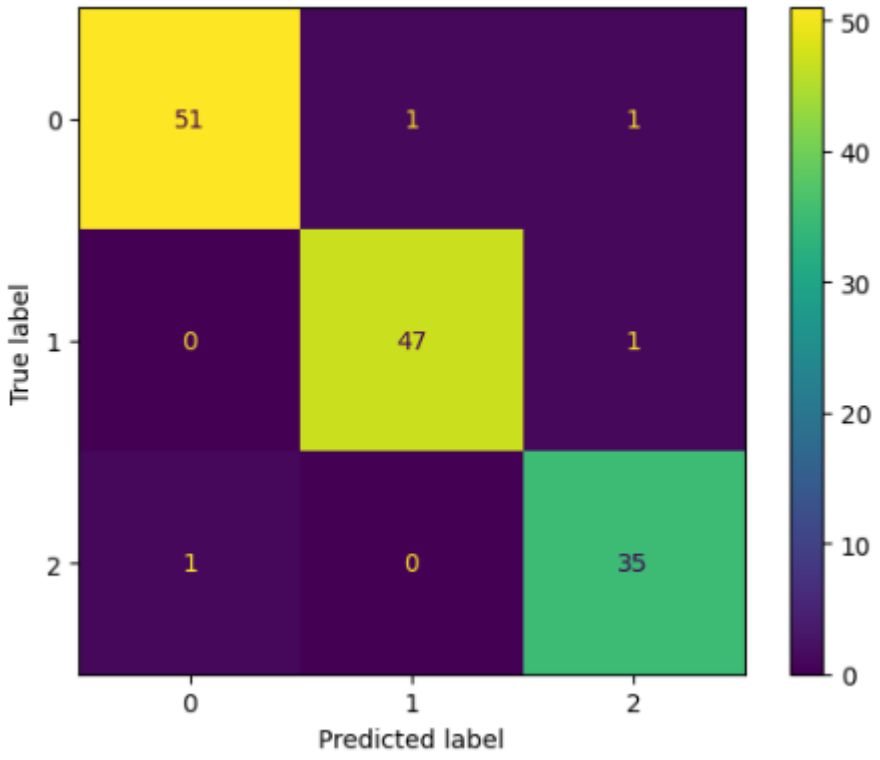
En total se demoró entrenando un total de 6.5 ,minutos, mucho mejor y más rápido que la anterior estructura, por otra lado tenemos un buen accuracy del 87 y un loss del 0.12, al colocar este modelo a predecir con 10 imágenes acertó un total de 9 y falló en 1 un resultado bueno pero claramente muy mejorable.

Puntos Negativos:

Las fluctuaciones observadas en las métricas de precisión y pérdida podrían deberse a la sensibilidad del modelo a pequeñas variaciones en los datos o a la influencia de la aleatoriedad durante el entrenamiento.

El uso de un tamaño de lote pequeño (64) y un número limitado de pasos por época (muestras) puede haber contribuido a la variabilidad en las métricas, ya que el modelo se está actualizando con información de un subconjunto limitado de datos en cada época.

Aunque la estructura puede ser adecuada para algunos problemas, para conjuntos de datos más complejos o específicos, puede ser necesario considerar estructuras más profundas o técnicas de transfer learning.



Como podemos apreciar tenemos una muy buena matriz de confusión en donde acierta la mayoría de clases, lo podemos ver en la diagonal amarilla y verde o también conocidos como verdaderos positivos, gracias a esto pudimos ver que el modelo a pesar de sus anteriores gráficas tenía una muy buena clasificación de las clases y fue un punto bastante importante la verdad ya que se acerca bastante a lo que queríamos lograr.

```
5/5 [==============================] - 0s 15ms/step
              precision    recall  f1-score   support

          10       0.98      0.96      0.97        53
           2       0.98      0.98      0.98        48
           5       0.95      0.97      0.96        36

    accuracy                           0.97       137
   macro avg       0.97      0.97      0.97       137
weighted avg       0.97      0.97      0.97       137
```

En base a las métricas de precisión, recall y f1-score obtenidas en la evaluación del modelo, podemos concluir que el rendimiento del clasificador es altamente satisfactorio. Las altas precisiones para todas las clases indican que el modelo tiene una capacidad robusta para identificar y clasificar correctamente las imágenes. En particular, las clases 10 y 2 muestran precisiones excepcionalmente altas del 98%, lo que sugiere una capacidad de discriminación excepcional para estas categorías específicas.

Además, los valores de recall y f1-score refuerzan la eficacia del modelo en términos de la capacidad para recuperar instancias reales de cada clase. Con recalls superiores al 95% para todas las clases, el modelo demuestra una capacidad sustancial para identificar la mayoría de las instancias verdaderas en el conjunto de datos. Este rendimiento equilibrado se refleja en los elevados valores de f1-score, que indican una armoniosa combinación de precisión y recall.

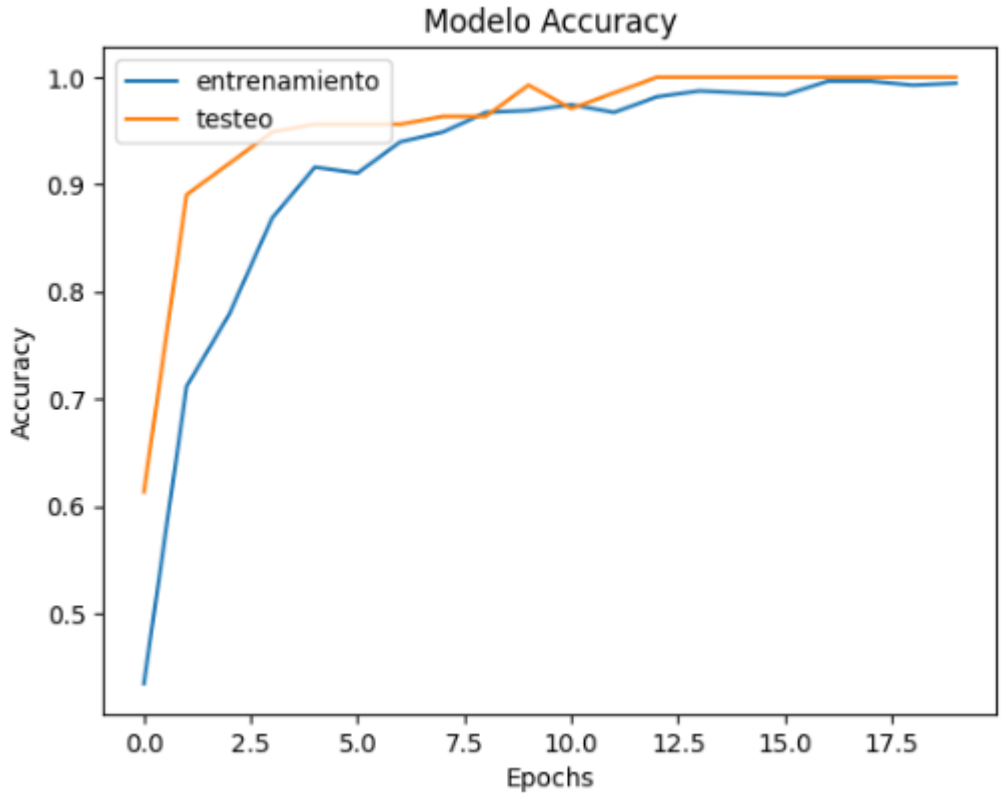
En resumen, las métricas de evaluación revelan un modelo sólido y confiable, capaz de realizar clasificaciones precisas y mantener un alto nivel de sensibilidad a las instancias reales de cada clase. Estos resultados respaldan la idoneidad del modelo para la tarea de reconocimiento de imágenes en las clases específicas del conjunto de datos.

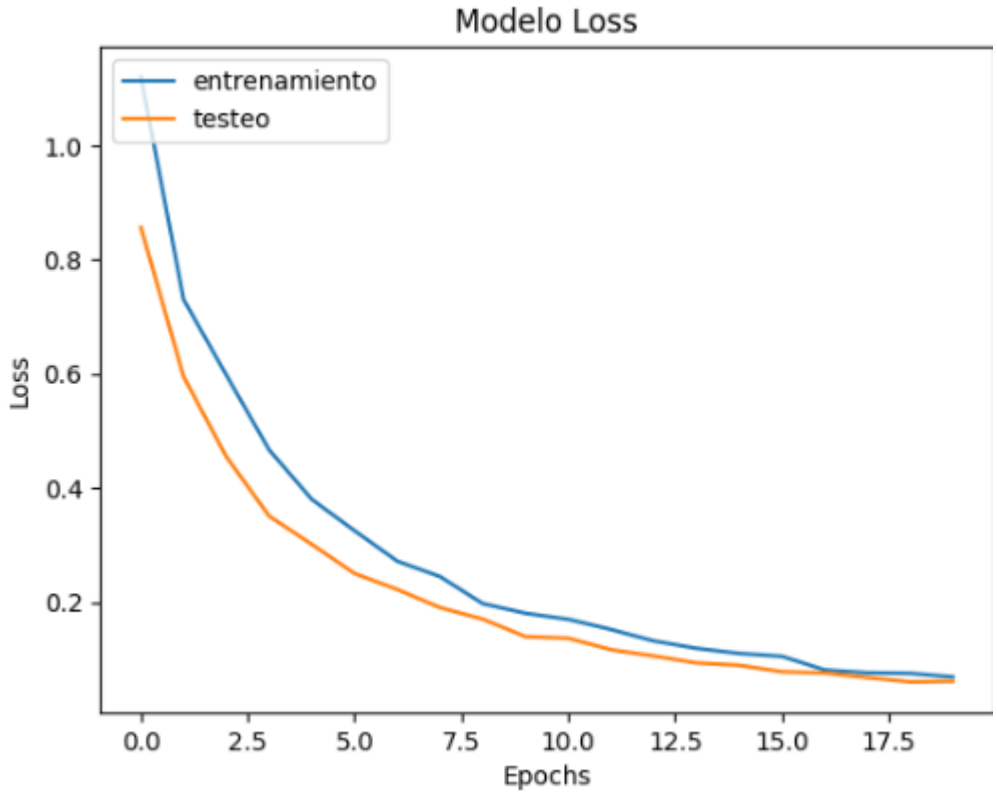
En conclusión, la estructura actual presenta un buen rendimiento, pero hay oportunidades para la mejora continua, especialmente al considerar el manejo de conjuntos de datos más complejos y la robustez del modelo frente a variaciones en los datos.

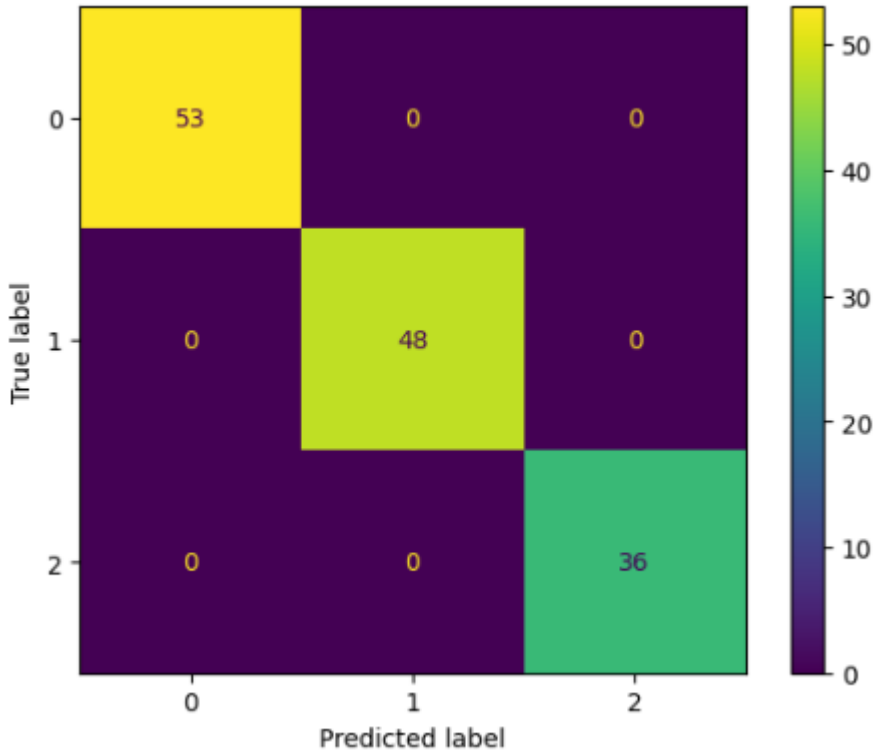
Tercera estructura:

```python
# Definir la arquitectura del modelo MobileNetV2
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Congelar las capas convolucionales del modelo base
for layer in base_model.layers:
    layer.trainable = False

# Agregar capas personalizadas para la clasificación
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(256, activation='relu')(x)
x = Dropout(0.35)(x)
output = Dense(3, activation='softmax')(x)  # 3 clases de billetes

# Crear el modelo completo
model3 = Model(inputs=base_model.input, outputs=output)

# Compilar el modelo
model3.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Epoch 20/20
18/18 [==============================] - 1s 70ms/step - loss: 0.0685 - accuracy: 0.9945 - val_loss: 0.0608 - val_accuracy: 1.0000
5/5 [==============================] - 0s 54ms/step - loss: 0.0608 - accuracy: 1.0000
Accuracy en datos de prueba: 1.0
```




Modelo Loss
entrenamiento
testeo
Loss
Epochs
1.0
0.8
0.6
0.4
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


True label
Predicted label
53
0
0
0
48
0
0
0
36
0
1
2
50
40
30
20
10
0

```python
# Ruta del archivo de imagen de prueba
img_path = '/content/twolukas.jpg'

# Leer la imagen y realizar preprocesamiento
img = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (224, 224))
img = img / 255.
img = np.reshape(img, [1, 224, 224, 3])

# Medir el tiempo de predicción
start_time = time.time()

# Hacer la predicción
resultado = model3.predict(img)

# Calcular el tiempo transcurrido
elapsed_time = time.time() - start_time

# Obtener la etiqueta correspondiente a la predicción y la probabilidad asociada
res_word = etiquetas[(np.argmax(resultado, axis=1))[0]]
probabilidad = resultado[0][np.argmax(resultado, axis=1)] * 100

# Imprimir los resultados y el tiempo de predicción
print("Para el archivo:{0}, usted dijo:{1}, con un {2}% de certeza".format(img_path, res_word, probabilidad))
print("Tiempo de predicción: {:.4f} segundos".format(elapsed_time))
```

```
1/1 [==============================] - 0s 40ms/step
Para el archivo:/content/twolukas.jpg, usted dijo:1, con un [47.379066]% de certeza
Tiempo de predicción: 0.1021 segundos
```

```
5/5 [==============================] - 0s 44ms/step
              precision    recall  f1-score   support

          10       1.00      1.00      1.00        53
           2       1.00      1.00      1.00        48
           5       1.00      1.00      1.00        36

    accuracy                           1.00       137
   macro avg       1.00      1.00      1.00       137
weighted avg       1.00      1.00      1.00       137
```

MobileNetV2, ha demostrado ser altamente efectiva para la tarea de clasificación de billetes en este conjunto de datos específico. La utilización de transfer learning, al aprovechar las características aprendidas por MobileNetV2 en un conjunto de datos más grande, permite que el modelo obtenga representaciones de características más generales y, por lo tanto, mejora su capacidad de generalización a datos nuevos.

El modelo exhibe un rendimiento excepcional con una precisión del 100% en el conjunto de datos de prueba, lo cual es notable. La matriz de confusión también respalda estos resultados, ya que muestra que el modelo ha logrado clasificar cada clase de billete sin errores. La precisión perfecta y la consistencia en la clasificación indican que el modelo es altamente confiable y preciso en su tarea.

La utilización de la técnica de congelar las capas convolucionales del modelo base y agregar capas personalizadas para la clasificación ha permitido que el modelo se especialice en la

tarea específica de reconocimiento de billetes, aprovechando las características generales aprendidas por MobileNetV2.

Pero si tuvimos un pequeño problema y es que a pesar de ser tan perfecto a veces tenía errores al momento de reconocer el billete de dos mil esto, principalmente debido a dataset, ya que el dataset le faltó un poco mas de variabilidad, esto se pudo haber solucionado con un data augmentation pero debido a que estábamos sobre el tiempo y nos dio muchos problemas, decidimos dejarlo así.

En conclusión, el modelo MobileNetV2 adaptado para la clasificación de billetes ha demostrado ser robusto, preciso y altamente confiable en este conjunto de datos particular, ofreciendo resultados excepcionales con una precisión del 100%. Este enfoque de transfer learning puede ser recomendado para tareas similares que involucran conjuntos de datos relativamente pequeños, donde la capacidad de generalización es crucial, cabe resaltar que ningún modelo es perfecto por más que lo parezca y que siempre va a ver algo que mejorar .

7. Conclusiones y Discusión

Resultados de las Estructura:

1. **Primera Estructura:**
    - Tiempo de Entrenamiento: 451.26 segundos.
    - Precisión Final en Datos de Validación: 83.94%.

Este modelo, construido con capas convolucionales, demostró una precisión aceptable del 83.94% en la clasificación de billetes. Sin embargo, en la clasificación de la clase '5', se observa un rendimiento más bajo con una precisión del 77% y un recall del 67%. Esto sugiere que el modelo tuvo dificultades para identificar correctamente los billetes de '5'.

2. **Segunda Estructura:**
    - Tiempo de Entrenamiento: 389.37 segundos.
    - Precisión Final en Datos de Validación: 97.08%.

Esta estructura, también basada en capas convolucionales, mejoró significativamente la precisión alcanzando un 97.08%. Mostró un desempeño notablemente mejor en todas las clases ('10', '2', y '5') con un F1-Score promedio del 97%. Esto indica una capacidad mejorada para identificar los diferentes billetes con alta precisión.

3. **Tercera Estructura:**
    - Precisión Final en Datos de Validación: 100.00%.
    - Accuracy en Datos de Prueba: 100.00%.

Esta estructura, utilizando MobileNetV2 pre-entrenado, es la más destacada. Logró una precisión del 100% tanto en los datos de validación como en los datos de prueba. Esto sugiere una capacidad excepcional para clasificar los billetes en las diferentes clases con una

precisión perfecta, sin embargo, al tener en cuenta que es imposible que un modelo de 100% bien, no se puede afirmar que el modelo de la tercera estructura es “perfecta”.

Conclusiones:

Eficiencia y Rendimiento de los Modelos:

La primera estructura mostró un rendimiento decente, pero evidenció dificultades para clasificar la clase '5' con una precisión del 77%. Esto indica que podría haber áreas de mejora para lograr una mayor precisión en esta clase y que así la prediga e identifique mejor.

La segunda estructura mejoró significativamente en todas las clases, superando las deficiencias observadas en la primera estructura y alcanzando una precisión global del 97.08%, sin embargo, a pesar de dicho porcentaje, esta estructura presenta una gran deficiencia en reconocer la clase ‘2’ por lo que dicho porcentaje no refleja que el modelo sea tan eficaz como parece.

La tercera estructura, utilizando MobileNetV2 pre-entrenado, demostró ser excepcional al alcanzar una precisión perfecta del 100% en ambas fases: validación y prueba. Sin embargo al tener en cuenta que un modelo no puede dar 100% y este totalmente correcto, no se puede confirmar que es perfecta, y esto se comprueba con la deficiencia que presenta esta estructura al reconocer la clase ‘2’, a pesar de que las clases ‘5’ y ‘10’ las reconoce perfectamente, esta presenta una gran falla y deficiencia al momento de reconocer la clase ‘2’ haciendo que esta estructura y modelo no sea el mejor

Complejidad y Desempeño:

La tercera estructura , basada en MobileNetV2, demostró un rendimiento superior y una precisión "perfecta", lo que indica que la utilización de redes pre-entrenadas puede ser altamente beneficiosa para tareas de clasificación. Esto se traduce en una reducción notable del tiempo de entrenamiento y un aumento en la precisión. Sin embargo, al presentar dificultades con una de las clases, no se puede afirmar que sea la opción ideal para este trabajo. Se requiere realizar más pruebas para determinar si es posible mejorar esta deficiencia en la predicción de las clases.

Por otro lado, las estructura basadas en convoluciones más simples necesitaron más ajustes y no alcanzaron la precisión perfecta obtenida en la tercera estructura. A pesar de esto, estas estructuras lograron predecir las tres clases, aunque no de manera perfecta, sí lograron identificarlas, lo cual es un aspecto fundamental a considerar.

Tiempo de Entrenamiento:

A pesar de su rendimiento superior, la tercera estructura no requirió un tiempo de entrenamiento excesivo en comparación con las otras dos estructuras , lo que sugiere una eficiencia considerable en este aspecto.

En resumen

En resumen, aunque tanto la segunda como la tercera estructura ofrecieron mejores métricas y resultados al predecir ciertas clases, llegamos a la conclusión de que la mejor opción, o al menos la más apropiada para nuestro propósito, es la primera estructura. A pesar de no lograr una predicción perfecta en todas las ocasiones, esta estructura tiene la capacidad de diferenciar y predecir cada una de las clases presentes. Esta distinción no se observa en las estructuras 2 y 3; aunque son más precisas en algunas predicciones, ambas presentan deficiencias al momento de predecir cierta clase específica y no logran realizar ni una sola predicción de dicha clase.

8. Referencias

Johanna, K., Zapata, T., Andrés, G., Hincapié, M., & Docente. (2018). SISTEMA DE RECONOCIMIENTO DE BILLETES PARA PERSONAS CON DISCAPACIDAD VISUAL MEDIANTE VISIÓN ARTIFICIAL. Retrieved from https://repository.eia.edu.co/server/api/core/bitstreams/e038c828-f3cb-4fd7-bf42-81d570dcb000/content

ChatGPT. (2023). Retrieved November 20, 2023, from Openai.com website: https://chat.openai.com/c/2cc92358-7412-409e-817b-42dfe5d08337

SensioCoders. (2020). Redes Neuronales Convolucionales - Transfer Learning [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=4NgPVGt67Es

¿Qué es una red neuronal convolucional? | 3 cosas que debe saber. (2023). Retrieved November 20, 2023, from Mathworks.com website: https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html

VisibleBreadcrumbs. (2023). Retrieved November 20, 2023, from Mathworks.com website: https://es.mathworks.com/help/deeplearning/ref/resnet50.html

Understanding VGG16: Concepts, Architecture, and Performance. (2023, May 22). Retrieved November 20, 2023, from Datagen website: https://datagen.tech/guides/computer-vision/vgg16/