**Documento: MobileNetV2 – Explicación, Funcionamiento y Uso**

## 1️⃣ Introducción

**MobileNetV2** es una arquitectura de red neuronal profunda desarrollada por Google, diseñada para **dispositivos móviles y aplicaciones con recursos limitados**. Su objetivo principal es **proporcionar un alto rendimiento en clasificación de imágenes y reconocimiento visual** utilizando **menos parámetros y menor costo computacional**, manteniendo buena precisión.

Se utiliza ampliamente en: - Clasificación de imágenes. - Detección de objetos. - Segmentación de imágenes. - Aplicaciones móviles con IA.

**Referencias oficiales:** - Paper original MobileNetV2: <https://arxiv.org/abs/1801.04381> - Documentación TensorFlow: <https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV2>

## 2️⃣ Cómo funciona MobileNetV2

### 🔹 Arquitectura

1. **Depthwise Separable Convolutions**:
   * Divide la convolución estándar en dos pasos:
     1. **Depthwise Convolution:** aplica un filtro a cada canal de la imagen.
     2. **Pointwise Convolution (1x1):** combina la información de los canales.
   * Reduce drásticamente la cantidad de parámetros.
2. **Inverted Residuals + Linear Bottleneck**
   * Aumenta la dimensionalidad antes de la convolución y luego la reduce (bottleneck lineal).
   * Retiene características importantes usando menos operaciones.

### 🔹 Flujo de datos

Input Image → Conv → Depthwise Separable Blocks → Bottleneck Residuals → GAP → Dense (Softmax) → Output

* **GAP:** Global Average Pooling, resume información antes de la clasificación.

## 3️⃣ Ventajas de MobileNetV2

| Ventaja | Explicación |
| --- | --- |
| Ligero | Menos parámetros que ResNet o VGG, ideal para móviles. |
| Rápido | Baja latencia, funciona bien en tiempo real. |
| Transfer Learning | Puede reutilizarse para nuevos datasets sin entrenar desde cero. |
| Precisión competitiva | Mantiene buena exactitud comparado con modelos más pesados. |

## 4️⃣ Uso práctico en TensorFlow/Keras

### 🔹 Cargar MobileNetV2 preentrenado

from tensorflow.keras.applications import MobileNetV2  
  
base\_model = MobileNetV2(input\_shape=(224,224,3), include\_top=False, weights='imagenet')

### 🔹 Agregar capas personalizadas

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D  
from tensorflow.keras.models import Model  
  
x = base\_model.output  
x = GlobalAveragePooling2D()(x)  
x = Dense(128, activation='relu')(x)  
output = Dense(3, activation='softmax')(x) # 3 clases  
  
model = Model(inputs=base\_model.input, outputs=output)

### 🔹 Compilar el modelo

from tensorflow.keras.optimizers import Adam  
  
model.compile(  
 optimizer=Adam(learning\_rate=0.0001),  
 loss='categorical\_crossentropy',  
 metrics=['accuracy']  
)

### 🔹 Entrenamiento con Data Augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
train\_datagen = ImageDataGenerator(  
 rescale=1./255,  
 rotation\_range=20,  
 width\_shift\_range=0.2,  
 height\_shift\_range=0.2,  
 horizontal\_flip=True,  
 zoom\_range=0.2  
)  
  
val\_datagen = ImageDataGenerator(rescale=1./255)  
  
train\_generator = train\_datagen.flow\_from\_directory('dataset/train', target\_size=(224,224), batch\_size=32, class\_mode='categorical')  
val\_generator = val\_datagen.flow\_from\_directory('dataset/val', target\_size=(224,224), batch\_size=32, class\_mode='categorical')  
  
history = model.fit(train\_generator, validation\_data=val\_generator, epochs=15)

## 5️⃣ Transfer Learning y Fine-Tuning

* **Congelar la base:** base\_model.trainable = False
* **Fine-tuning:** desbloquear algunas capas para mejorar la adaptación al dataset:

base\_model.trainable = True  
for layer in base\_model.layers[:100]:  
 layer.trainable = False

## 6️⃣ Guardar y cargar el modelo

# Guardar  
model.save('modelo\_final.h5')  
  
# Cargar  
from tensorflow.keras.models import load\_model  
model = load\_model('modelo\_final.h5')

## 7️⃣ Recursos y enlaces útiles

* Paper MobileNetV2 (arXiv): <https://arxiv.org/abs/1801.04381>
* Keras Applications: <https://www.tensorflow.org/api_docs/python/tf/keras/applications>
* Tutorial Transfer Learning con MobileNetV2: <https://www.tensorflow.org/tutorials/images/transfer_learning>
* Código de ejemplo GitHub: <https://github.com/tensorflow/models>