Fine-tuning code using **MobileNetV2** in **TensorFlow/Keras**, designed for a **binary image classification task** like **cats vs dogs**.

✅ What this does:

- Loads pre-trained MobileNetV2.
- Adds new layers for your task.
- Trains those new layers first.
- Then fine-tunes the entire model.

📦 Requirements

Install TensorFlow if you haven't:

```
pip install tensorflow
```

You'll also need a directory like:

```
data/
└── cats_and_dogs/
    ├── cats/
    │   ├── cat1.jpg
    │   └── ...
    └── dogs/
        ├── dog1.jpg
        └── ...
```

✅ Full Code

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

# --------------------
# 1. Load Pretrained Base Model
# --------------------
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False  # Freeze base layers initially

# --------------------
# 2. Add Custom Top Layers
# --------------------
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)  # For binary classification

model = Model(inputs=base_model.input, outputs=output)

# --------------------
# 3. Compile Model (initial training)
# --------------------
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# --------------------
# 4. Prepare Data
# --------------------
data_dir = 'data/cats_and_dogs'

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    horizontal_flip=True,
    zoom_range=0.2
```

```
)

train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

# --------------------
# 5. Train Top Layers Only
# --------------------
print("Training new classification head...")
model.fit(train_generator, epochs=5, validation_data=val_generator)

# --------------------
# 6. Fine-tune the Whole Model
# --------------------
base_model.trainable = True  # Unfreeze the base model

# Set a lower learning rate for fine-tuning
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
      loss='binary_crossentropy',
      metrics=['accuracy'])

print("Fine-tuning entire model...")
model.fit(train_generator, epochs=5, validation_data=val_generator)

# --------------------
# 7. Save Model
# --------------------
model.save("fine_tuned_mobilenetv2.h5")
print("Model saved as fine_tuned_mobilenetv2.h5")
```

📝 **Notes**

- You can increase epochs as needed.
- You can use model.evaluate() to test on unseen data.
- If your dataset is not in folders, I can help you adapt this for CSV/image-label pairs.
