Model Improvement to avoid overfitting

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
train_dir = '/content/drive/My Drive/train'
test_dir = '/content/drive/My Drive/test1'
```

```
img_size = (150, 150)
batch_size = 128

# Data augmentation for training
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2  # 20% validation data
)

# Train and validation generators
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',  # Multi-class classification
    shuffle=True,
    subset='training'  # Training subset
)

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',  # Multi-class classification
    shuffle=False,
    subset='validation'  # Validation subset
)
```

```
Found 20000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
```

```
steps_per_epoch = np.ceil(train_generator.samples / batch_size).astype(int)
validation_steps = np.ceil(validation_generator.samples / batch_size).astype(int)
```

```
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
base_model.trainable = False  # Freeze the base model layers

# Build the model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),  # Dropout for regularization
    Dense(train_generator.num_classes, activation='softmax')  # Output layer
])
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_orderin
9406464/9406464 [==============================] - 0s 0us/step
```

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


early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_checkpoint = ModelCheckpoint(
    'best_mobilenetv2_model.keras', save_best_only=True, monitor='val_loss'
)




Image = model.fit(
    train_generator,
    validation_data=validation_generator,
    steps_per_epoch=steps_per_epoch,
    validation_steps=validation_steps,
    epochs=10,
    callbacks=[early_stopping, model_checkpoint],
    verbose=1
)
```

```
Epoch 1/10
157/157 [==============================] - 3419s 20s/step - loss: 0.2450 - accuracy: 0.8994 - val_loss: 0.1657 - val_accuracy: 0.925
Epoch 2/10
157/157 [==============================] - 230s 1s/step - loss: 0.1727 - accuracy: 0.9255 - val_loss: 0.1591 - val_accuracy: 0.9282
Epoch 3/10
157/157 [==============================] - 231s 1s/step - loss: 0.1625 - accuracy: 0.9316 - val_loss: 0.1526 - val_accuracy: 0.9320
Epoch 4/10
157/157 [==============================] - 230s 1s/step - loss: 0.1595 - accuracy: 0.9315 - val_loss: 0.1465 - val_accuracy: 0.9396
Epoch 5/10
157/157 [==============================] - 229s 1s/step - loss: 0.1586 - accuracy: 0.9331 - val_loss: 0.1525 - val_accuracy: 0.9350
Epoch 6/10
157/157 [==============================] - 231s 1s/step - loss: 0.1523 - accuracy: 0.9365 - val_loss: 0.1381 - val_accuracy: 0.9436
Epoch 7/10
157/157 [==============================] - 230s 1s/step - loss: 0.1464 - accuracy: 0.9359 - val_loss: 0.1410 - val_accuracy: 0.9408
Epoch 8/10
157/157 [==============================] - 230s 1s/step - loss: 0.1473 - accuracy: 0.9379 - val_loss: 0.1501 - val_accuracy: 0.9360
Epoch 9/10
157/157 [==============================] - 230s 1s/step - loss: 0.1525 - accuracy: 0.9353 - val_loss: 0.1474 - val_accuracy: 0.9390
Epoch 10/10
157/157 [==============================] - 229s 1s/step - loss: 0.1478 - accuracy: 0.9378 - val_loss: 0.1385 - val_accuracy: 0.9404
```

```
model.evaluate(validation_generator)
```

```
40/40 [==============================] - 44s 1s/step - loss: 0.1422 - accuracy: 0.9398
[0.1422402411699295, 0.9398000240325928]
```

```
model.save('final_mobilenetv2_model.h5')
```

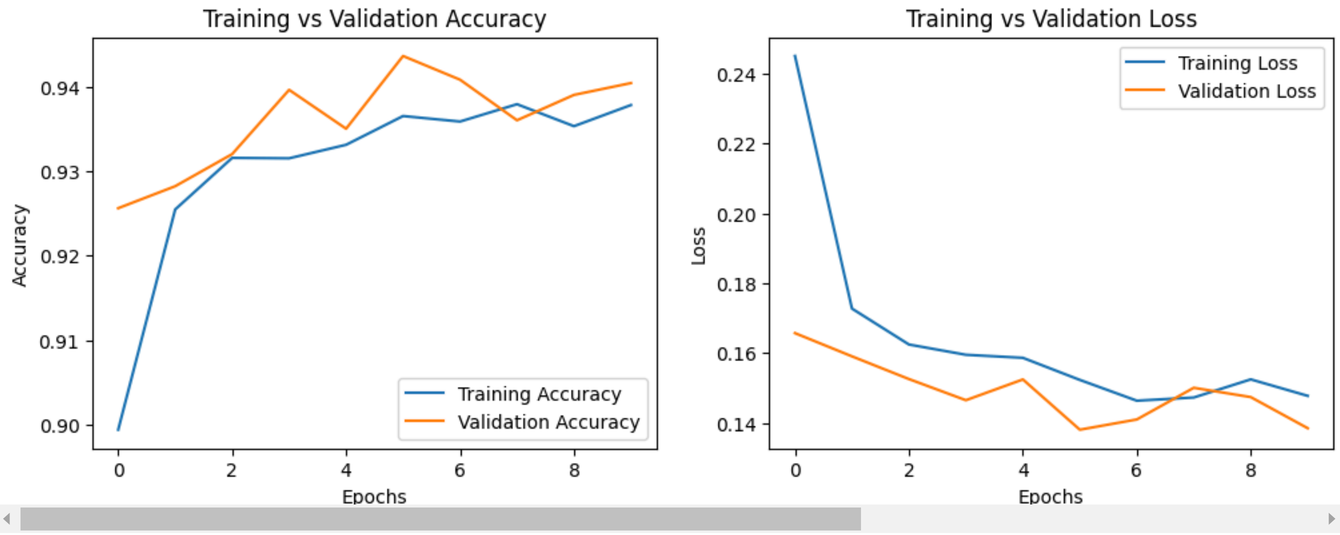
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file vi
  saving_api.save_model(
```

```
plt.figure(figsize=(12, 4))

# Plot accuracy
plt.subplot(1, 2, 1)
plt.plot(Image.history['accuracy'], label='Training Accuracy')
plt.plot(Image.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training vs Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(Image.history['loss'], label='Training Loss')
plt.plot(Image.history['val_loss'], label='Validation Loss')
plt.title('Training vs Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



```
import os
from IPython.display import display
import ipywidgets as widgets
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np

# Define the test directory and load the trained model
test_dir = '/content/drive/My Drive/test1'
model = model  # Use the model you trained earlier

# Use training generator to get class indices (ensure it's defined earlier in your code)
class_indices = train_generator.class_indices  # Get the class indices from the training generator
class_labels = list(class_indices.keys())  # List of class names

# Get all image filenames in the test directory
image_files = [f for f in os.listdir(test_dir) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

# Dropdown widget for selecting an image
dropdown = widgets.Dropdown(
    options=image_files,
    description='Select Image:',
    style={'description_width': 'initial'}
)

# Function to make predictions and display the selected image
def display_prediction(change):
    selected_image = change.new
    image_path = os.path.join(test_dir, selected_image)

    # Load and preprocess the image
    img = Image.open(image_path).resize((150, 150))  # Resize to match model input size
    img_array = np.array(img) / 255.0  # Normalize pixel values
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension

    # Make a prediction
    prediction = model.predict(img_array)

    if prediction.shape[1] == 1:  # Binary classification
        confidence = prediction[0][0]
        predicted_class = 'Dog' if confidence > 0.5 else 'Cat'
        confidence = confidence if predicted_class == 'Dog' else 1 - confidence
    else:  # Multi-class classification
        confidence = np.max(prediction)  # Confidence of the predicted class
        predicted_class_index = np.argmax(prediction)
        predicted_class = class_labels[predicted_class_index]  # Use training class labels

    # Display the image and prediction with confidence
    plt.imshow(img)
    plt.title(f"Prediction: {predicted_class} ({confidence:.2f})")
    plt.axis('off')
    plt.show()

# Attach the function to the dropdown
dropdown.observe(display_prediction, names='value')

# Display the dropdown
print("Select an image to verify the prediction:")
```

```
    display(dropdown)
```

```
Select an image to verify the prediction:
```

Select Image: 9214.jpg

```
1/1 [==============================] - 0s 40ms/step
```

Prediction: cat (1.00)

```
1/1 [==============================] - 0s 39ms/step
```

Prediction: dog (1.00)

```
1/1 [==============================] - 0s 43ms/step
```

Prediction: cat (1.00)

```
1/1 [==============================] - 0s 40ms/step
```

Prediction: dog (1.00)

Start coding or generate with AI.

```
!pip install pillow tkinterdnd2
```

```
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (11.0.0)
Collecting tkinterdnd2
  Downloading tkinterdnd2-0.4.2-py3-none-any.whl.metadata (2.7 kB)
Downloading tkinterdnd2-0.4.2-py3-none-any.whl (492 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 492.7/492.7 kB 7.2 MB/s eta 0:00:00
Installing collected packages: tkinterdnd2
Successfully installed tkinterdnd2-0.4.2
```

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.8.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.6-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.4.0-py3-none-any.whl.metadata (2.9 kB)
Collecting gradio-client==1.5.1 (from gradio)
  Downloading gradio_client-1.5.1-py3-none-any.whl.metadata (7.1 kB)
Collecting httpx>=0.24.1 (from gradio)
  Downloading httpx-0.28.1-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: huggingface-hub>=0.25.1 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.26.3)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.4)
Collecting markupsafe~=2.0 (from gradio)
  Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.26.4)
Collecting orjson~=3.0 (from gradio)
  Downloading orjson-3.10.12-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.8/41.8 kB 2.3 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (11.0.0)
Requirement already satisfied: pydantic>=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.10.3)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.19-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.2.2 (from gradio)
  Downloading ruff-0.8.2-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.41.3-py3-none-any.whl.metadata (6.0 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.15.0)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.12.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.32.1-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==1.5.1->gradio) (2024.10.0)
Collecting websockets<15.0,>=10.0 (from gradio-client==1.5.1->gradio)
  Downloading websockets-14.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.m
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (2024.8.30)
Collecting httpcore==1.* (from httpx>=0.24.1->gradio)
  Downloading httpcore-1.0.7-py3-none-any.whl.metadata (21 kB)
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx>=0.24.1->gradio)
  Downloading h11-0.14.0-py3-none-any.whl.metadata (8.2 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (3.16.1
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (2.32.3
```

```
import gradio as gr
import gradio as gr
import tensorflow as tf
from PIL import Image
import numpy as np

# Load the trained model
model = tf.keras.models.load_model('final_mobilenetv2_model.h5')  # Update with your model's path

# Preprocessing function
def preprocess_image(image):
    image = image.resize((150, 150))  # Resize to model input size
    img_array = np.array(image) / 255.0  # Normalize pixel values
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
```