```
#Animal Classification Model


import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.applications.efficientnet import preprocess_input
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau
import matplotlib.pyplot as plt
import numpy as np


# Define dataset path
dataset_dir = "dataset"  # Update with your dataset path

# Updated Image Preprocessing & Augmentation
data_gen = ImageDataGenerator(
rescale=1./255,
rotation_range=50,  # Increased for better variation
width_shift_range=0.3,
height_shift_range=0.3,
shear_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
fill_mode='nearest',
validation_split=0.2  # 80% training, 20% validation
)

# Load dataset
batch_size = 32
img_height = 224
img_width = 224

train_generator = data_gen.flow_from_directory(
dataset_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='categorical',
subset='training'
)

val_generator = data_gen.flow_from_directory(
dataset_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='categorical',
subset='validation'
)

# Load pre-trained EfficientNetB0 model
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(img_height, img_

# Unfreeze the last 20 layers for fine-tuning
for layer in base_model.layers[-20:]:
layer.trainable = True
```

```
# Build model
model = Sequential([
base_model,
GlobalAveragePooling2D(),
Dense(256, activation='relu'),
Dense(15, activation='softmax')  # 15 classes
])

# Learning Rate Scheduler
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1)

# Compile model
model.compile(optimizer=Adam(learning_rate=0.001),
loss='categorical_crossentropy',
metrics=['accuracy'])

# Train model
history = model.fit(train_generator, validation_data=val_generator, epochs=50, callbacks=[lr_scl

# Plot training history
def plot_history(history):
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plot_history(history)

# Save the trained model
model.save("animal_classification_model.h5")

print("Training Completed. Model saved as animal_classification_model.h5")
```