PROJECT TITLE:- HUMAN EMOTION CLASSIFICATION

PROJECT DESCRIPTION=

1.CNN

2.IMAGE AUGMENTATION

3.TRANSFER LEARNING

```
!unzip "/content/drive/MyDrive/COMPUTER VISION-SUKAMAL JYOTI DAS/COMPUTER VISION PROJECT/LUCI
```

```
      inflating: LUCIFER/train/happiness/images (2).jpg
      inflating: LUCIFER/train/happiness/images (3).jpg
      inflating: LUCIFER/train/happiness/images (4).jpg
      inflating: LUCIFER/train/happiness/images (5).jpg
      inflating: LUCIFER/train/happiness/images (6).jpg
      inflating: LUCIFER/train/happiness/images (7).jpg
      inflating: LUCIFER/train/happiness/images (8).jpg
      inflating: LUCIFER/train/happiness/images (9).jpg
      inflating: LUCIFER/train/happiness/images.jpg
      inflating: LUCIFER/train/neutrality/download (1).jpg
      inflating: LUCIFER/train/neutrality/download (10).jpg
      inflating: LUCIFER/train/neutrality/download (11).jpg
      inflating: LUCIFER/train/neutrality/download (12).jpg
      inflating: LUCIFER/train/neutrality/download (2).jpg
      inflating: LUCIFER/train/neutrality/download (3).jpg
      inflating: LUCIFER/train/neutrality/download (4).jpg
      inflating: LUCIFER/train/neutrality/download (5).jpg
      inflating: LUCIFER/train/neutrality/download (6).jpg
      inflating: LUCIFER/train/neutrality/download (7).jpg
      inflating: LUCIFER/train/neutrality/download (8).jpg
      inflating: LUCIFER/train/neutrality/download (9).jpg
      inflating: LUCIFER/train/neutrality/download.jpg
      inflating: LUCIFER/train/neutrality/images (1).jpg
      inflating: LUCIFER/train/neutrality/images (10).jpg
      inflating: LUCIFER/train/neutrality/images (11).jpg
      inflating: LUCIFER/train/neutrality/images (2).jpg
      inflating: LUCIFER/train/neutrality/images (3).jpg
      inflating: LUCIFER/train/neutrality/images (4).jpg
      inflating: LUCIFER/train/neutrality/images (5).jpg
      inflating: LUCIFER/train/neutrality/images (6).jpg
      inflating: LUCIFER/train/neutrality/images (7).jpg
      inflating: LUCIFER/train/neutrality/images (8).jpg
      inflating: LUCIFER/train/neutrality/images (9).jpg
      inflating: LUCIFER/train/neutrality/images.jpg
      inflating: LUCIFER/train/sadness/download (1).jpg
      inflating: LUCIFER/train/sadness/download (10).jpg
      inflating: LUCIFER/train/sadness/download (11).jpg
      inflating: LUCIFER/train/sadness/download (2).jpg
      inflating: LUCIFER/train/sadness/download (3).jpg
      inflating: LUCIFER/train/sadness/download (4).jpg
      inflating: LUCIFER/train/sadness/download (5).jpg
      inflating: LUCIFER/train/sadness/download (6).jpg
      inflating: LUCIFER/train/sadness/download (7).jpg
      inflating: LUCIFER/train/sadness/download (8).jpg
      inflating: LUCIFER/train/sadness/download (9).jpg
      inflating: LUCIFER/train/sadness/download.jpg
      inflating: LUCIFER/train/sadness/images (1).jpg
      inflating: LUCIFER/train/sadness/images (10).jpg
      inflating: LUCIFER/train/sadness/images (11).jpg
      inflating: LUCIFER/train/sadness/images (12).jpg
      inflating: LUCIFER/train/sadness/images (2).jpg
      inflating: LUCIFER/train/sadness/images (3).jpg
      inflating: LUCIFER/train/sadness/images (4).jpg
      inflating: LUCIFER/train/sadness/images (5).jpg
      inflating: LUCIFER/train/sadness/images (6).jpg
      inflating: LUCIFER/train/sadness/images (7).jpg
      inflating: LUCIFER/train/sadness/images (8).jpg
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf

from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import Dense, Flatten, Conv2D,Conv1D, MaxPooling2D,MaxPooling1D


#data
import os

base_dir = 'LUCIFER'

train_dir = 'LUCIFER/train'
test_dir = 'LUCIFER/test'

train_anger_dir ='LUCIFER/train/anger'
train_happiness_dir ='LUCIFER/train/happiness'
train_neutrality_dir ='LUCIFER/train/neutrality'
train_sadness_dir ='LUCIFER/train/sadness'


test_anger_dir ='LUCIFER/test/anger'
test_happiness_dir ='LUCIFER/test/happiness'
test_neutrality_dir ='LUCIFER/test/neutrality'
test_sadness_dir ='LUCIFER/test/sadness'


print('total train anger images:', len(os.listdir(train_anger_dir)))
print('total train happiness images:', len(os.listdir(train_happiness_dir)))
print('total train neutrality images:', len(os.listdir(train_neutrality_dir)))
print('total train sadness images:', len(os.listdir(train_sadness_dir)))
print('total test anger images:', len(os.listdir(test_anger_dir)))
print('total test happiness images:', len(os.listdir(test_happiness_dir)))
print('total test neutrality images:', len(os.listdir(test_neutrality_dir)))
print('total test sadness images:', len(os.listdir(test_sadness_dir)))
```

```
total train anger images: 25
total train happiness images: 25
total train neutrality images: 25
total train sadness images: 25
total test anger images: 25
total test happiness images: 25
total test neutrality images: 25
total test sadness images: 25
```

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu',input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 74, 74, 32)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 36, 36, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 34, 34, 128)       73856     
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 17, 17, 128)      0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 15, 15, 128)       147584    
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 7, 7, 128)        0         
 2D)                                                             
 flatten_3 (Flatten)         (None, 6272)              0

 dense_6 (Dense)             (None, 512)               3211776

 dense_7 (Dense)             (None, 4)                 2052

=================================================================
Total params: 3,454,660
Trainable params: 3,454,660
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,target_size=(150,150),batch_siz
test_generator = test_datagen.flow_from_directory(test_dir,target_size=(150,150),batch_size=3
```

```
Found 100 images belonging to 4 classes.
Found 100 images belonging to 4 classes.
```

```
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
tf.__version__
```

```
'2.7.0'
```

```
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])
#model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])
model.fit_generator(train_generator,epochs=2)
```

```
Epoch 1/2
4/4 [==============================] - 6s 1s/step - loss: 1.6769 - accuracy: 0.2300
Epoch 2/2
4/4 [==============================] - 5s 1s/step - loss: 1.3911 - accuracy: 0.2400
<keras.callbacks.History at 0x7fb137dd0850>
```

```
test_loss, test_accuracy = model.evaluate(test_generator)
```

```
4/4 [==============================] - 2s 308ms/step - loss: 1.3822 - accuracy: 0.2500
```

```
print(test_loss)
print(test_accuracy)
```

```
1.3822414875030518
0.25
```

Image Augmentation

```
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(train_dir,target_size=(150,150),batch_siz
test_generator = test_datagen.flow_from_directory(test_dir,target_size=(150,150),batch_size=3
```

```
Found 100 images belonging to 4 classes.
Found 100 images belonging to 4 classes.
```

```
model.compile(loss="categorical_crossentropy",optimizer="rmsprop",metrics=['accuracy'])
model.fit_generator(train_generator,epochs=5)
```

```
Epoch 1/5
4/4 [==============================] - 6s 1s/step - loss: 1.4224 - accuracy: 0.2300
Epoch 2/5
4/4 [==============================] - 5s 1s/step - loss: 1.4059 - accuracy: 0.2500
Epoch 3/5
4/4 [==============================] - 5s 2s/step - loss: 1.3889 - accuracy: 0.2700
Epoch 4/5
4/4 [==============================] - 5s 2s/step - loss: 1.3879 - accuracy: 0.2600
Epoch 5/5
4/4 [==============================] - 5s 1s/step - loss: 1.3901 - accuracy: 0.2300
<keras.callbacks.History at 0x7fb1396b6c10>
```

Transfer Learning

```
from tensorflow.keras.applications import VGG16
```

```
conv_base = VGG16(weights="imagenet", include_top=False, input_shape=(150,150,3))#imagenet
```

```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 input_3 (InputLayer)        [(None, 150, 150, 3)]     0

 block1_conv1 (Conv2D)       (None, 150, 150, 64)      1792

 block1_conv2 (Conv2D)       (None, 150, 150, 64)      36928

 block1_pool (MaxPooling2D)  (None, 75, 75, 64)        0

 block2_conv1 (Conv2D)       (None, 75, 75, 128)       73856

 block2_conv2 (Conv2D)       (None, 75, 75, 128)       147584

 block2_pool (MaxPooling2D)  (None, 37, 37, 128)       0

 block3_conv1 (Conv2D)       (None, 37, 37, 256)       295168

 block3_conv2 (Conv2D)       (None, 37, 37, 256)       590080

 block3_conv3 (Conv2D)       (None, 37, 37, 256)       590080

 block3_pool (MaxPooling2D)  (None, 18, 18, 256)       0

 block4_conv1 (Conv2D)       (None, 18, 18, 512)       1180160

 block4_conv2 (Conv2D)       (None, 18, 18, 512)       2359808

 block4_conv3 (Conv2D)       (None, 18, 18, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 9, 9, 512)         0

 block5_conv1 (Conv2D)       (None, 9, 9, 512)         2359808

 block5_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

 block5_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

```
tl_model = Sequential()

tl_model.add(conv_base)

tl_model.add(Flatten())
tl_model.add(Dense(512, activation='relu'))
tl_model.add(Dense(4, activation='softmax'))
tl_model.summary()
```

```
conv_base.trainable = False
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 flatten_4 (Flatten)         (None, 8192)              0

 dense_8 (Dense)             (None, 512)               4194816

 dense_9 (Dense)             (None, 4)                 2052

=================================================================
Total params: 18,911,556
Trainable params: 18,911,556
Non-trainable params: 0
```

```
tl_model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])
#tl_model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])
tl_model.fit_generator(train_generator,epochs=15)
```

```
Epoch 1/15
4/4 [==============================] - 24s 7s/step - loss: 4.2847 - accuracy: 0.2200
Epoch 2/15
4/4 [==============================] - 23s 5s/step - loss: 2.5695 - accuracy: 0.3100
Epoch 3/15
4/4 [==============================] - 23s 7s/step - loss: 2.0088 - accuracy: 0.3100
Epoch 4/15
4/4 [==============================] - 24s 8s/step - loss: 1.3358 - accuracy: 0.4900
Epoch 5/15
4/4 [==============================] - 23s 5s/step - loss: 1.2343 - accuracy: 0.4700
Epoch 6/15
4/4 [==============================] - 23s 5s/step - loss: 1.1385 - accuracy: 0.5300
Epoch 7/15
4/4 [==============================] - 23s 5s/step - loss: 0.7997 - accuracy: 0.7200
Epoch 8/15
4/4 [==============================] - 23s 5s/step - loss: 1.0268 - accuracy: 0.6000
Epoch 9/15
4/4 [==============================] - 23s 5s/step - loss: 0.9255 - accuracy: 0.6700
Epoch 10/15
4/4 [==============================] - 23s 5s/step - loss: 1.0366 - accuracy: 0.5900
Epoch 11/15
4/4 [==============================] - 23s 5s/step - loss: 0.9643 - accuracy: 0.5400
Epoch 12/15
4/4 [==============================] - 23s 5s/step - loss: 0.8616 - accuracy: 0.6300
Epoch 13/15
4/4 [==============================] - 23s 7s/step - loss: 1.0569 - accuracy: 0.5500
Epoch 14/15
4/4 [==============================] - 23s 5s/step - loss: 0.7372 - accuracy: 0.6800
Epoch 15/15
4/4 [==============================] - 23s 5s/step - loss: 0.7445 - accuracy: 0.7400
<keras.callbacks.History at 0x7fb1b85c50d0>
```

```
test_loss, test_accuracy = tl_model.evaluate(test_generator)
print(test_loss)
print(test_accuracy)
```

```
     4/4 [==============================] - 23s 5s/step - loss: 0.3973 - accuracy: 0.8900
     0.3973011374473572
     0.8899999856948853
```

```
# Freezing all layers upto a specific one

conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False


tl_model.compile(loss='categorical_crossentropy', optimizer="adam",metrics=['accuracy'])
tl_model.fit_generator(train_generator,epochs=15)
```

```
     Epoch 1/15
     4/4 [==============================] - 29s 6s/step - loss: 36.1865 - accuracy: 0.3700
     Epoch 2/15
     4/4 [==============================] - 28s 6s/step - loss: 1.5012 - accuracy: 0.2200
     Epoch 3/15
     4/4 [==============================] - 28s 6s/step - loss: 2.1704 - accuracy: 0.2800
     Epoch 4/15
     4/4 [==============================] - 28s 6s/step - loss: 1.6743 - accuracy: 0.2200
     Epoch 5/15
     4/4 [==============================] - 28s 6s/step - loss: 1.4253 - accuracy: 0.2600
     Epoch 6/15
     4/4 [==============================] - 28s 6s/step - loss: 1.3846 - accuracy: 0.2800
     Epoch 7/15
     4/4 [==============================] - 28s 9s/step - loss: 2.0209 - accuracy: 0.2100
     Epoch 8/15
     4/4 [==============================] - 28s 6s/step - loss: 1.3851 - accuracy: 0.2000
     Epoch 9/15
     4/4 [==============================] - 27s 6s/step - loss: 1.4957 - accuracy: 0.2600
     Epoch 10/15
     4/4 [==============================] - 28s 6s/step - loss: 2.7106 - accuracy: 0.2800
     Epoch 11/15
     4/4 [==============================] - 27s 9s/step - loss: 1.4190 - accuracy: 0.3500
     Epoch 12/15
     4/4 [==============================] - 27s 6s/step - loss: 2.3120 - accuracy: 0.2700
     Epoch 13/15
     4/4 [==============================] - 27s 6s/step - loss: 1.4583 - accuracy: 0.2800
     Epoch 14/15
```

```
4/4 [==============================] - 27s 6s/step - loss: 1.4213 - accuracy: 0.2700
Epoch 15/15
4/4 [==============================] - 27s 6s/step - loss: 1.3863 - accuracy: 0.2900
<keras.callbacks.History at 0x7fb1b8507450>
```

```
from tensorflow.keras.applications import ResNet50
```

```
conv_base = ResNet50(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
```

```
conv_base.summary()
```

```
 conv5_block1_add (Add)         (None, 5, 5, 2048)   0           ['conv5_block1_0_bn[
                                                                  'conv5_block1_3_bn[

 conv5_block1_out (Activation)  (None, 5, 5, 2048)   0           ['conv5_block1_add[0

 conv5_block2_1_conv (Conv2D)   (None, 5, 5, 512)    1049088     ['conv5_block1_out[0

 conv5_block2_1_bn (BatchNormal  (None, 5, 5, 512)   2048        ['conv5_block2_1_conv
 ization)

 conv5_block2_1_relu (Activatio  (None, 5, 5, 512)   0           ['conv5_block2_1_bn[
 n)

 conv5_block2_2_conv (Conv2D)   (None, 5, 5, 512)    2359808     ['conv5_block2_1_rel

 conv5_block2_2_bn (BatchNormal  (None, 5, 5, 512)   2048        ['conv5_block2_2_conv
 ization)

 conv5_block2_2_relu (Activatio  (None, 5, 5, 512)   0           ['conv5_block2_2_bn[
 n)

 conv5_block2_3_conv (Conv2D)   (None, 5, 5, 2048)   1050624     ['conv5_block2_2_rel

 conv5_block2_3_bn (BatchNormal  (None, 5, 5, 2048)  8192        ['conv5_block2_3_conv
 ization)

 conv5_block2_add (Add)         (None, 5, 5, 2048)   0           ['conv5_block1_out[0
                                                                  'conv5_block2_3_bn[

 conv5_block2_out (Activation)  (None, 5, 5, 2048)   0           ['conv5_block2_add[0

 conv5_block3_1_conv (Conv2D)   (None, 5, 5, 512)    1049088     ['conv5_block2_out[0

 conv5_block3_1_bn (BatchNormal  (None, 5, 5, 512)   2048        ['conv5_block3_1_conv
 ization)

 conv5_block3_1_relu (Activatio  (None, 5, 5, 512)   0           ['conv5_block3_1_bn[
 n)

 conv5_block3_2_conv (Conv2D)   (None, 5, 5, 512)    2359808     ['conv5_block3_1_rel

 conv5_block3_2_bn (BatchNormal  (None, 5, 5, 512)   2048        ['conv5_block3_2_conv
 ization)
 conv5_block3_2_relu (Activatio  (None, 5, 5, 512)   0           ['conv5_block3_2_bn[
 n)

 conv5_block3_3_conv (Conv2D)   (None, 5, 5, 2048)   1050624     ['conv5_block3_2_rel

 conv5_block3_3_bn (BatchNormal  (None, 5, 5, 2048)  8192        ['conv5_block3_3_con
 ization)

 conv5_block3_add (Add)         (None, 5, 5, 2048)   0           ['conv5_block2_out[0
                                                                  'conv5_block3_3_bn[

 conv5_block3_out (Activation)  (None, 5, 5, 2048)   0           ['conv5_block3_add[0

=======================================================================================
```

```
tl_model = Sequential()
tl_model.add(conv_base)
tl_model.add(Flatten())
tl_model.add(Dense(512, activation='relu'))
tl_model.add(Dense(4, activation='softmax'))

conv_base.trainable = False


tl_model.compile(loss='categorical_crossentropy',optimizer="adam",metrics=['accuracy'])
tl_model.fit_generator(train_generator,epochs=15)
```

```
Epoch 1/15
4/4 [==============================] - 13s 3s/step - loss: 10.8007 - accuracy: 0.2500
Epoch 2/15
4/4 [==============================] - 9s 3s/step - loss: 4.3706 - accuracy: 0.3000
Epoch 3/15
4/4 [==============================] - 9s 3s/step - loss: 3.3633 - accuracy: 0.2700
Epoch 4/15
4/4 [==============================] - 9s 3s/step - loss: 2.0629 - accuracy: 0.2700
Epoch 5/15
4/4 [==============================] - 9s 2s/step - loss: 1.7012 - accuracy: 0.2600
Epoch 6/15
4/4 [==============================] - 9s 3s/step - loss: 1.5786 - accuracy: 0.2800
Epoch 7/15
4/4 [==============================] - 9s 2s/step - loss: 1.4076 - accuracy: 0.3500
Epoch 8/15
4/4 [==============================] - 9s 3s/step - loss: 1.4512 - accuracy: 0.2500
Epoch 9/15
4/4 [==============================] - 9s 2s/step - loss: 1.3935 - accuracy: 0.3200
Epoch 10/15
4/4 [==============================] - 9s 3s/step - loss: 1.4334 - accuracy: 0.2300
Epoch 11/15
4/4 [==============================] - 9s 2s/step - loss: 1.3386 - accuracy: 0.3700
Epoch 12/15
4/4 [==============================] - 9s 2s/step - loss: 1.3394 - accuracy: 0.3400
Epoch 13/15
4/4 [==============================] - 9s 2s/step - loss: 1.4328 - accuracy: 0.2700
```

```
Epoch 14/15
4/4 [==============================] - 9s 2s/step - loss: 1.5011 - accuracy: 0.2700
Epoch 15/15
4/4 [==============================] - 9s 2s/step - loss: 1.4265 - accuracy: 0.3700
<keras.callbacks.History at 0x7fb13926c050>
```

```
test_loss,·test_accuracy·=·tl_model.evaluate(test_generator)
print(test_loss)
print(test_accuracy)
```

```
4/4 [==============================] - 9s 2s/step - loss: 1.3834 - accuracy: 0.2700
1.383420705795288
0.27000001072883606
```

```
# Freezing all layers upto a specific one

conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

```
tl_model.compile(loss='categorical_crossentropy',optimizer="adam",metrics=['accuracy'])
tl_model.fit_generator(train_generator,epochs=15)
```

```
Epoch 1/15
4/4 [==============================] - 13s 2s/step - loss: 2.2688 - accuracy: 0.3000
Epoch 2/15
4/4 [==============================] - 9s 2s/step - loss: 1.9927 - accuracy: 0.2500
Epoch 3/15
4/4 [==============================] - 9s 3s/step - loss: 1.8611 - accuracy: 0.2500
Epoch 4/15
4/4 [==============================] - 9s 2s/step - loss: 1.4612 - accuracy: 0.2900
Epoch 5/15
4/4 [==============================] - 9s 2s/step - loss: 1.4693 - accuracy: 0.2800
Epoch 6/15
4/4 [==============================] - 9s 2s/step - loss: 1.4485 - accuracy: 0.2600
Epoch 7/15
4/4 [==============================] - 9s 3s/step - loss: 1.5287 - accuracy: 0.2900
Epoch 8/15
4/4 [==============================] - 11s 3s/step - loss: 1.4272 - accuracy: 0.2600
Epoch 9/15
4/4 [==============================] - 9s 3s/step - loss: 1.4618 - accuracy: 0.2500
Epoch 10/15
4/4 [==============================] - 9s 2s/step - loss: 1.3533 - accuracy: 0.3100
Epoch 11/15
4/4 [==============================] - 9s 2s/step - loss: 1.4009 - accuracy: 0.2900
```

```
    Epoch 12/15
    4/4 [==============================] - 9s 2s/step - loss: 1.3509 - accuracy: 0.2900
    Epoch 13/15
    4/4 [==============================] - 9s 2s/step - loss: 1.3607 - accuracy: 0.3000
    Epoch 14/15
    4/4 [==============================] - 9s 2s/step - loss: 1.3418 - accuracy: 0.3200
    Epoch 15/15
    4/4 [==============================] - 9s 2s/step - loss: 1.3767 - accuracy: 0.3200
    <keras.callbacks.History at 0x7fb137cda1d0>
```

```
test_loss, test_accuracy = tl_model.evaluate(test_generator)
print(test_loss)
print(test_accuracy)
```

```
    4/4 [==============================] - 9s 2s/step - loss: 1.3052 - accuracy: 0.3800
    1.3052204847335815
    0.3799999952316284
```

✓ 9s completed at 2:44 AM