## 1. Data Augmentation

```
# Import necessary lib.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Data augmentation on training variable

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

# Data augmentation on testing variable

test_datagen = ImageDataGenerator(rescale=1./255)

# Data augmentation on training data

xtrain =
train_datagen.flow_from_directory('./Animal_Dataset/dataset/Training',
 target_size=(64,64),
 class_mode='categorical',
 batch_size=100)
```

```
Found 1238 images belonging to 4 classes.
```

```
# Data augmentation on testing data

xtest =
test_datagen.flow_from_directory('./Animal_Dataset/dataset/Testing',
 target_size=(64,64),
 class_mode='categorical',
 batch_size=100)
```

```
Found 326 images belonging to 4 classes.
```

## 2.CNN model training

```
# Importing req. lib.

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D,
Flatten, Dense

# Build a CNN block

model = Sequential() # Initializing sequential model
model.add(Convolution2D(32,
(3,3),activation='relu',input_shape=(64,64,3))) # convolution layer
model.add(MaxPooling2D(pool_size=(2, 2))) # Max pooling layer
model.add(Flatten()) # Flatten layer
```

```
model.add(Dense(300,activation='relu')) # Hidden layer 1
model.add(Dense(150,activation='relu')) # Hidden layer 2
model.add(Dense(4,activation='softmax')) # Output layer

# Compiling the model

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

```
13
```

```
# Train model

model.fit_generator(xtrain,
 steps_per_epoch=len(xtrain),
 epochs=25,
 validation_data=xtest,
 validation_steps=len(xtest))
```

```
C:\Users\Arjun\AppData\Local\Temp\ipykernel_10028\3802424551.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  model.fit_generator(xtrain,
```

```
Epoch 1/25
13/13 [==============================] - 46s 3s/step - loss: 1.7762 -
accuracy: 0.2666 - val_loss: 1.2531 - val_accuracy: 0.3497
Epoch 2/25
13/13 [==============================] - 70s 6s/step - loss: 1.2015 -
accuracy: 0.4443 - val_loss: 1.0715 - val_accuracy: 0.6319
Epoch 3/25
13/13 [==============================] - 72s 6s/step - loss: 1.0355 -
accuracy: 0.5767 - val_loss: 1.0134 - val_accuracy: 0.6227
Epoch 4/25
13/13 [==============================] - 57s 4s/step - loss: 0.8678 -
accuracy: 0.6648 - val_loss: 0.7236 - val_accuracy: 0.7423
Epoch 5/25
13/13 [==============================] - 56s 4s/step - loss: 0.7786 -
accuracy: 0.6826 - val_loss: 0.6909 - val_accuracy: 0.7301
Epoch 6/25
13/13 [==============================] - 50s 4s/step - loss: 0.6868 -
accuracy: 0.7464 - val_loss: 0.5578 - val_accuracy: 0.8037
Epoch 7/25
13/13 [==============================] - 59s 5s/step - loss: 0.6103 -
accuracy: 0.7787 - val_loss: 0.4966 - val_accuracy: 0.8098
Epoch 8/25
13/13 [==============================] - 72s 6s/step - loss: 0.6940 -
accuracy: 0.7302 - val_loss: 0.5413 - val_accuracy: 0.8190
Epoch 9/25
13/13 [==============================] - 59s 5s/step - loss: 0.6058 -
```

```
accuracy: 0.7754 - val_loss: 0.4215 - val_accuracy: 0.8497
Epoch 10/25
13/13 [==============================] - 63s 5s/step - loss: 0.5429 -
accuracy: 0.7908 - val_loss: 0.5823 - val_accuracy: 0.7699
Epoch 11/25
13/13 [==============================] - 59s 5s/step - loss: 0.5150 -
accuracy: 0.8069 - val_loss: 0.3818 - val_accuracy: 0.8804
Epoch 12/25
13/13 [==============================] - 92s 7s/step - loss: 0.4805 -
accuracy: 0.8029 - val_loss: 0.3960 - val_accuracy: 0.8436
Epoch 13/25
13/13 [==============================] - 89s 7s/step - loss: 0.4145 -
accuracy: 0.8473 - val_loss: 0.3819 - val_accuracy: 0.8558
Epoch 14/25
13/13 [==============================] - 61s 5s/step - loss: 0.3581 -
accuracy: 0.8691 - val_loss: 0.2216 - val_accuracy: 0.9294
Epoch 15/25
13/13 [==============================] - 58s 4s/step - loss: 0.3624 -
accuracy: 0.8578 - val_loss: 0.4041 - val_accuracy: 0.8405
Epoch 16/25
13/13 [==============================] - 78s 6s/step - loss: 0.3464 -
accuracy: 0.8683 - val_loss: 0.2000 - val_accuracy: 0.9387
Epoch 17/25
13/13 [==============================] - 96s 7s/step - loss: 0.3211 -
accuracy: 0.8821 - val_loss: 0.3073 - val_accuracy: 0.8834
Epoch 18/25
13/13 [==============================] - 98s 8s/step - loss: 0.3048 -
accuracy: 0.8796 - val_loss: 0.2325 - val_accuracy: 0.9110
Epoch 19/25
13/13 [==============================] - 75s 6s/step - loss: 0.2640 -
accuracy: 0.9111 - val_loss: 0.4452 - val_accuracy: 0.8190
Epoch 20/25
13/13 [==============================] - 65s 5s/step - loss: 0.2253 -
accuracy: 0.9265 - val_loss: 0.1335 - val_accuracy: 0.9325
Epoch 21/25
13/13 [==============================] - 54s 4s/step - loss: 0.2513 -
accuracy: 0.9039 - val_loss: 0.1600 - val_accuracy: 0.9540
Epoch 22/25
13/13 [==============================] - 51s 4s/step - loss: 0.2390 -
accuracy: 0.9168 - val_loss: 0.1651 - val_accuracy: 0.9509
Epoch 23/25
13/13 [==============================] - 56s 4s/step - loss: 0.1962 -
accuracy: 0.9386 - val_loss: 0.1442 - val_accuracy: 0.9448
Epoch 24/25
13/13 [==============================] - 60s 5s/step - loss: 0.1440 -
accuracy: 0.9588 - val_loss: 0.1158 - val_accuracy: 0.9601
Epoch 25/25
13/13 [==============================] - 59s 5s/step - loss: 0.1341 -
accuracy: 0.9572 - val_loss: 0.0638 - val_accuracy: 0.9847
```

```
<keras.callbacks.History at 0x275325375b0>
```

```
# Save model

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

## 3.Testing model

```
from tensorflow.keras.preprocessing import image
import numpy as np

img =
image.load_img('./Animal_Dataset/dataset/Testing//elephants/mala_mala_
200064__340.jpg',target_size=(600,600))

img
```

```
x = image.img_to_array(img)

x

array([[[253., 253., 253.],
        [255., 255., 253.],
        [254., 254., 252.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [254., 254., 255.]],

       [[253., 253., 253.],
        [255., 255., 253.],
        [254., 254., 252.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [254., 254., 255.]],

       [[254., 255., 255.],
        [245., 247., 244.],
        [254., 255., 253.],
        ...,
        [253., 253., 253.],
        [255., 255., 255.],
        [255., 255., 255.]],

       ...,

       [[159., 173., 138.],
        [170., 184., 151.],
        [164., 177., 147.],
        ...,
        [157., 170., 152.],
        [127., 138., 121.],
        [142., 151., 134.]],

       [[149., 163., 128.],
        [149., 163., 128.],
        [134., 148., 115.],
        ...,
        [143., 156., 136.],
        [136., 148., 128.],
        [142., 151., 132.]],

       [[149., 163., 128.],
        [149., 163., 128.],
        [134., 148., 115.],
        ...,
```

```
        [143., 156., 136.],
        [136., 148., 128.],
        [142., 151., 132.]]], dtype=float32)
```

```
img = np.expand_dims(x,axis=0)
```

```
img
```

```
array([[[[253., 253., 253.],
         [255., 255., 253.],
         [254., 254., 252.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [254., 254., 255.]],

        [[253., 253., 253.],
         [255., 255., 253.],
         [254., 254., 252.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [254., 254., 255.]],

        [[254., 255., 255.],
         [245., 247., 244.],
         [254., 255., 253.],
         ...,
         [253., 253., 253.],
         [255., 255., 255.],
         [255., 255., 255.]],

        ...,

        [[159., 173., 138.],
         [170., 184., 151.],
         [164., 177., 147.],
         ...,
         [157., 170., 152.],
         [127., 138., 121.],
         [142., 151., 134.]],

        [[149., 163., 128.],
         [149., 163., 128.],
         [134., 148., 115.],
         ...,
         [143., 156., 136.],
         [136., 148., 128.],
         [142., 151., 132.]],
```

```
[[149., 163., 128.],
 [149., 163., 128.],
 [134., 148., 115.],
 ...,
 [143., 156., 136.],
 [136., 148., 128.],
 [142., 151., 132.]]], dtype=float32)
```