```
pwd
```

```
'/home/wsuser/work'
```

```
!pip install keras==2.2.4
!pip install tensorflow==1.14.0
```

```
Collecting keras==2.2.4
  Downloading Keras-2.2.4-py2.py3-none-any.whl (312 kB)
ent already satisfied: pyyaml in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
keras==2.2.4) (5.4.1)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
keras==2.2.4) (1.1.2)
Collecting keras-applications>=1.0.6
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
ent already satisfied: numpy>=1.9.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
keras==2.2.4) (1.20.3)
Requirement already satisfied: h5py in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
keras==2.2.4) (3.2.1)
Requirement already satisfied: scipy>=0.14 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from keras==2.2.4) (1.7.3)
Requirement already satisfied: six>=1.9.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from keras==2.2.4) (1.15.0)
Installing collected packages: keras-applications, keras
  Attempting uninstall: keras
    Found existing installation: keras 2.7.0
    Uninstalling keras-2.7.0:
      Successfully uninstalled keras-2.7.0
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
tensorflow 2.7.2 requires keras<2.8,>=2.7.0, but you have keras 2.2.4
which is incompatible.
Successfully installed keras-2.2.4 keras-applications-1.0.8
ERROR: Could not find a version that satisfies the requirement
tensorflow==1.14.0 (from versions: 2.5.0, 2.5.1, 2.5.2, 2.5.3,
2.6.0rc0, 2.6.0rc1, 2.6.0rc2, 2.6.0, 2.6.1, 2.6.2, 2.6.3, 2.6.4,
2.6.5, 2.7.0rc0, 2.7.0rc1, 2.7.0, 2.7.1, 2.7.2, 2.7.3, 2.7.4,
2.8.0rc0, 2.8.0rc1, 2.8.0, 2.8.1, 2.8.2, 2.8.3, 2.9.0rc0, 2.9.0rc1,
2.9.0rc2, 2.9.0, 2.9.1, 2.9.2, 2.10.0rc0, 2.10.0rc1, 2.10.0rc2,
2.10.0rc3, 2.10.0, 2.11.0rc0, 2.11.0rc1, 2.11.0rc2)
ERROR: No matching distribution found for tensorflow==1.14.0
```

```
import os, types
import pandas as pd
from botocore.client import Config
```

```
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage.
It includes your credentials.
# You might want to remove those credentials before you share the
notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='N01idcdE0afRzWu54iY_Gv6n6XmG0AGVu9-xTisCRyF1',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-
storage.appdomain.cloud')

bucket = 'imageclasification-donotdelete-pr-duouh4elvoeftg'
object_key = 'Animal_Dataset.zip'
streaming_body_1= cos_client.get_object(Bucket=bucket, Key=object_key)
['Body']

# Your data file was loaded into a botocore.response.StreamingBody
object.
# Please read the documentation of ibm_boto3 and pandas to learn more
about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/

!unzip streaming_body_1

unzip:  cannot find or open streaming_body_1, streaming_body_1.zip or
streaming_body_1.ZIP.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

Using TensorFlow backend.

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

test_datagen = ImageDataGenerator(rescale=1./255)



}
xtrain = train_datagen.flow_from_directory('streaming_body_1',
                                           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('/content/dataset/Testing',
                                         target_size=(64,64),
                                         class_mode='categorical',
                                         batch_size=100)
```

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

## CNN model training

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

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

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

model.fit_generator(xtrain,
                    steps_per_epoch=len(xtrain),
                    epochs=10,
                    validation_data=xtest,
                    validation_steps=len(xtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  import sys

Epoch 1/10
13/13 [==============================] - 18s 490ms/step - loss: 1.8822
- accuracy: 0.2908 - val_loss: 1.2656 - val_accuracy: 0.3558
Epoch 2/10
13/13 [==============================] - 6s 493ms/step - loss: 1.2039
- accuracy: 0.4661 - val_loss: 1.0490 - val_accuracy: 0.6319
Epoch 3/10
13/13 [==============================] - 6s 473ms/step - loss: 1.0134
- accuracy: 0.5969 - val_loss: 0.9052 - val_accuracy: 0.6135
Epoch 4/10
```

```
13/13 [==============================] - 6s 471ms/step - loss: 0.8838
- accuracy: 0.6470 - val_loss: 0.7871 - val_accuracy: 0.6442
Epoch 5/10
13/13 [==============================] - 6s 462ms/step - loss: 0.7732
- accuracy: 0.6963 - val_loss: 0.6946 - val_accuracy: 0.7423
Epoch 6/10
13/13 [==============================] - 6s 467ms/step - loss: 0.6976
- accuracy: 0.7229 - val_loss: 0.6000 - val_accuracy: 0.7669
Epoch 7/10
13/13 [==============================] - 6s 470ms/step - loss: 0.6408
- accuracy: 0.7585 - val_loss: 0.4942 - val_accuracy: 0.8405
Epoch 8/10
13/13 [==============================] - 6s 473ms/step - loss: 0.5708
- accuracy: 0.7779 - val_loss: 0.5511 - val_accuracy: 0.7945
Epoch 9/10
13/13 [==============================] - 6s 466ms/step - loss: 0.5394
- accuracy: 0.7981 - val_loss: 0.5997 - val_accuracy: 0.8006
Epoch 10/10
13/13 [==============================] - 6s 459ms/step - loss: 0.5215
- accuracy: 0.7948 - val_loss: 0.4270 - val_accuracy: 0.8620

<keras.callbacks.History at 0x7f57a224e290>
```

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

## Testing model

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

img =
image.load_img('/content/dataset/Testing/elephants/photo_1552055570_5c
41ef975579.jpeg',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
img =
image.load_img('/content/dataset/Testing/bears/m10.jpeg',target_size=(
64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
# Testing 4

img = image.load_img('/content/dataset/Testing/rats/images
(55).jpeg',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
xtrain.class_indices
```

```
{'bears': 0, 'crows': 1, 'elephants': 2, 'rats': 3}
```

## Model tuning

```
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau

early_stop = EarlyStopping(monitor='val_accuracy',
                           patience=5)

lr = ReduceLROnPlateau(monitor='val_accuaracy',
                       factor=0.5,
                       min_lr=0.00001)

callback = [early_stop,lr]

# Train model

model.fit_generator(xtrain,
                    steps_per_epoch=len(xtrain),
                    epochs=100,
                    callbacks=callback,
                    validation_data=xtest,
                    validation_steps=len(xtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.


Epoch 1/100
13/13 [==============================] - ETA: 0s - loss: 0.5002 -
accuracy: 0.8174
```

```
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 503ms/step - loss: 0.5002
- accuracy: 0.8174 - val_loss: 0.3022 - val_accuracy: 0.8988 - lr:
0.0010
Epoch 2/100
13/13 [==============================] - ETA: 0s - loss: 0.4280 -
accuracy: 0.8409

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 458ms/step - loss: 0.4280
- accuracy: 0.8409 - val_loss: 0.4246 - val_accuracy: 0.8466 - lr:
0.0010
Epoch 3/100
13/13 [==============================] - ETA: 0s - loss: 0.4018 -
accuracy: 0.8522

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 469ms/step - loss: 0.4018
- accuracy: 0.8522 - val_loss: 0.3677 - val_accuracy: 0.8773 - lr:
0.0010
Epoch 4/100
13/13 [==============================] - ETA: 0s - loss: 0.3703 -
accuracy: 0.8675

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 472ms/step - loss: 0.3703
- accuracy: 0.8675 - val_loss: 0.2848 - val_accuracy: 0.8681 - lr:
0.0010
Epoch 5/100
13/13 [==============================] - ETA: 0s - loss: 0.3317 -
accuracy: 0.8821

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 472ms/step - loss: 0.3317
- accuracy: 0.8821 - val_loss: 0.2316 - val_accuracy: 0.9110 - lr:
0.0010
Epoch 6/100
```

```
13/13 [==============================] - ETA: 0s - loss: 0.2919 -
accuracy: 0.8982

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 463ms/step - loss: 0.2919
- accuracy: 0.8982 - val_loss: 0.2037 - val_accuracy: 0.9141 - lr:
0.0010
Epoch 7/100
13/13 [==============================] - ETA: 0s - loss: 0.2653 -
accuracy: 0.9168

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 472ms/step - loss: 0.2653
- accuracy: 0.9168 - val_loss: 0.3426 - val_accuracy: 0.8681 - lr:
0.0010
Epoch 8/100
13/13 [==============================] - ETA: 0s - loss: 0.2562 -
accuracy: 0.9087

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 470ms/step - loss: 0.2562
- accuracy: 0.9087 - val_loss: 0.1796 - val_accuracy: 0.9264 - lr:
0.0010
Epoch 9/100
13/13 [==============================] - ETA: 0s - loss: 0.2512 -
accuracy: 0.9192

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 489ms/step - loss: 0.2512
- accuracy: 0.9192 - val_loss: 0.2118 - val_accuracy: 0.9080 - lr:
0.0010
Epoch 10/100
13/13 [==============================] - ETA: 0s - loss: 0.2113 -
accuracy: 0.9386

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
```

```
 13/13 [==============================] - 6s 470ms/step - loss: 0.2113
- accuracy: 0.9386 - val_loss: 0.2592 - val_accuracy: 0.9141 - lr:
0.0010
Epoch 11/100
13/13 [==============================] - ETA: 0s - loss: 0.1966 -
accuracy: 0.9443

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 458ms/step - loss: 0.1966
- accuracy: 0.9443 - val_loss: 0.1606 - val_accuracy: 0.9387 - lr:
0.0010
Epoch 12/100
13/13 [==============================] - ETA: 0s - loss: 0.1956 -
accuracy: 0.9297

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 454ms/step - loss: 0.1956
- accuracy: 0.9297 - val_loss: 0.1876 - val_accuracy: 0.9325 - lr:
0.0010
Epoch 13/100
13/13 [==============================] - ETA: 0s - loss: 0.2050 -
accuracy: 0.9330

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 467ms/step - loss: 0.2050
- accuracy: 0.9330 - val_loss: 0.1395 - val_accuracy: 0.9356 - lr:
0.0010
Epoch 14/100
13/13 [==============================] - ETA: 0s - loss: 0.1471 -
accuracy: 0.9548

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 456ms/step - loss: 0.1471
- accuracy: 0.9548 - val_loss: 0.1050 - val_accuracy: 0.9601 - lr:
0.0010
Epoch 15/100
13/13 [==============================] - ETA: 0s - loss: 0.1326 -
accuracy: 0.9564
```

```
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 478ms/step - loss: 0.1326
- accuracy: 0.9564 - val_loss: 0.0418 - val_accuracy: 0.9969 - lr:
0.0010
Epoch 16/100
13/13 [==============================] - ETA: 0s - loss: 0.1432 -
accuracy: 0.9556

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 459ms/step - loss: 0.1432
- accuracy: 0.9556 - val_loss: 0.1110 - val_accuracy: 0.9724 - lr:
0.0010
Epoch 17/100
13/13 [==============================] - ETA: 0s - loss: 0.1214 -
accuracy: 0.9669

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 472ms/step - loss: 0.1214
- accuracy: 0.9669 - val_loss: 0.0772 - val_accuracy: 0.9785 - lr:
0.0010
Epoch 18/100
13/13 [==============================] - ETA: 0s - loss: 0.0967 -
accuracy: 0.9750

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 455ms/step - loss: 0.0967
- accuracy: 0.9750 - val_loss: 0.0904 - val_accuracy: 0.9693 - lr:
0.0010
Epoch 19/100
13/13 [==============================] - ETA: 0s - loss: 0.1162 -
accuracy: 0.9637

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 511ms/step - loss: 0.1162
- accuracy: 0.9637 - val_loss: 0.0409 - val_accuracy: 0.9908 - lr:
0.0010
Epoch 20/100
```

```
13/13 [==============================] - ETA: 0s - loss: 0.0937 -
accuracy: 0.9717

WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuaracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr

 13/13 [==============================] - 6s 468ms/step - loss: 0.0937
- accuracy: 0.9717 - val_loss: 0.0448 - val_accuracy: 0.9877 - lr:
0.0010

<keras.callbacks.History at 0x7f57902a8490>

# Testing 4

img = image.load_img('/content/dataset/Testing/rats/images
(55).jpeg',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output

{"type":"string"}

# Testing google image

img =
image.load_img('/content/42ny6cwj8t_Polar_bear_on_ice_in_Svalbard_Norw
ay_WW294883.jpg',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output

{"type":"string"}

# Testing google image

img = image.load_img('/content/images.jpg',target_size=(64,64)) #
Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output

{"type":"string"}
```

```
# Testing google image

img = image.load_img('/content/01-rat-friends-
nationalgeographic_1162144_16x9.jpg',target_size=(64,64)) # Reading
image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```

```
# Testing google image

img = image.load_img('/content/photo-1599921778557-
082147629542.jpg',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity
index
op = ['bears','crows','elephants','rats'] # Creating list
op[pred] # List indexing with output
```

```
{"type":"string"}
```