Name: Marcela Pantoja

Class: 4375.004

Image Classification with DL

```
import tensorflow as tf
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
num_images = 0

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
        num_images = num_images +1
```

```
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/08108fb36d20a11d.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/3ac48c2576b04de9.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/03c7118d468d94aa.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/13626a8bd4c27b49.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/00bb45a8568a3474.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/4d5ff3bad0444108.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/048b18e2e692a424.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/3054d68c25d49d54.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/007f9ffc01c7b106.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/9d6b7b5bb7d5efcd.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/0321448f2848ab99.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/07642e275bb2b869.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/b2dda5f3f0bdad53.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/c8f2e212394ed401.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/223fc16f8023cf96.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/02a5846a35629f1d.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/4201c8c35efbfb2a.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/012f5ecbf49e1da0.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/03f680447efaccfe.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/8b48ed579956d124.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/767e3c2968ff1890.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/b3d2f85dee085875.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/db21d2f3774862b0.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/02dd8dd4344b04a7.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/2f6bb1f322b6d58a.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/c041a3ef0923aad5.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/029b39c0b65ebeb3.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/0ae4f7841c26ba0d.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/05632fadcb5e1e9c.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/1ebe26a6abe51106.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/341bedaae1e7ad89.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/113fbf2203ccef57.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/4a15bd602934d7aa.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/52b64d96fc0647e8.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/0db6601682a368e8.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/01d688c043bdbfbb.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/1da9b32b6e25e64c.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/7f33de86429cef7f.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/02b086c4e96396f6.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/0fb021bac7207533.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/0c61b8d86a3e0889.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/75ea322880cfcc3a.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/06434b4d5989df37.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/2546b57e18682a68.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/1bea54a4c02d2829.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/a9be4e0ac3ff7f4e.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/ed51aa4321f10a21.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/8f34ab7a9b7a8f9b.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/039d20705ddc6162.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/00c8d36882dd6d37.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/09017fd0f60f309a.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/2fd0d44cf194d16c.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/c65aae5dd1d75984.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/135b67b72eb7bb47.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/784940fed90cccc2.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/9a470427f0a875d2.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/519587ae1f0b5160.jpg
/kaggle/input/lions-or-cheetahs-image-classification/images/Cheetahs/00707659aba29334.jpg
```

```
print("Number of images in the data set: ", num_images)
```

```
Number of images in the data set:  200
```

Divide the Dataset into train/test split

```
#Adding a bunch of necessary packages needed for seperating the data set.
#Already have come from the beginning
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

#We're going to be splitting the data set into a 80/20 split, so with 200 images it should be 160/40.
#Theres only two classes, cheeta and lion

train_img = tf.keras.utils.image_dataset_from_directory('/kaggle/input/lions-or-cheetahs-image-classification/images', validation_split=0.2, subset="trainin

val_img = tf.keras.utils.image_dataset_from_directory('/kaggle/input/lions-or-cheetahs-image-classification/images', validation_split=0.2, subset="validatio

val_batches = tf.data.experimental.cardinality(val_img)
test_seg = val_img.take(val_batches // 5)
val_img = val_img.skip(val_batches // 5)
```

```
Found 200 files belonging to 2 classes.
Using 160 files for training.
Found 200 files belonging to 2 classes.
Using 40 files for validation.
```

Graph showing the distribution of the target classes

```
plt.figure(figsize=(6,5))
plt.xticks(np.arange(2))
plt.title("Distribution of Target Classes")
plt.xlabel("Target Classes")
plt.ylabel("Num of Images")
plt.bar(['Cheetas', 'Lions'], 100)
plt.show()
```

The data set I have chosen contains and even number of images of cheetas and lions. The model I am creating should be able to give an accurate prediction of whether or not the image shown is a cheeta of a lion. There are images of cheetas and lions in the data set in a variety of different positions and areas in an effort to train the model better.

Note: I had to make the image sizes very large in order to get the pictures to show up as legible, I experimented for a while with the image sizes. There are also a number of images that seem to be netiher a lion or a cheeta, these have a tendency to be in the lion class though.

Create a sequential model and evaluate on the test data

```
#Ok, so here is the model building part

model = tf.keras.models.Sequential([
  tf.keras.layers.Rescaling(1./499),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(500, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(500, activation='relu'),
```

```
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(2, activation='softmax')
])



#Here is the fixing the build and making sure its good
model.build(input_shape=(None, 125, 125, 3))

#Outputting the summary of the model befor compiling
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_7 (Rescaling)     (None, 125, 125, 3)       0

 flatten_7 (Flatten)         (None, 46875)             0

 dense_21 (Dense)            (None, 500)               23438000

 dropout_14 (Dropout)        (None, 500)               0

 dense_22 (Dense)            (None, 500)               250500

 dropout_15 (Dropout)        (None, 500)               0

 dense_23 (Dense)            (None, 2)                 1002

=================================================================
Total params: 23,689,502
Trainable params: 23,689,502
Non-trainable params: 0
_________________________________________________________________
```

```
#Model compilation
#Using optimizer Adamax at first loss and metric=accuracy of course
model.compile(optimizer='Adamax', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

#8 epochs to start, maybe change if bad accuracy
history = model.fit(train_img, validation_data=val_img, epochs=8)
```

```
Epoch 1/8
7/7 [==============================] - 4s 340ms/step - loss: 10.6832 - accuracy: 0.5562 - val_loss: 9.3781 - val_accuracy: 0.4500
Epoch 2/8
7/7 [==============================] - 3s 318ms/step - loss: 5.8222 - accuracy: 0.4875 - val_loss: 5.1514 - val_accuracy: 0.4500
Epoch 3/8
7/7 [==============================] - 3s 315ms/step - loss: 3.2184 - accuracy: 0.5250 - val_loss: 1.9272 - val_accuracy: 0.5500
Epoch 4/8
7/7 [==============================] - 3s 318ms/step - loss: 2.4875 - accuracy: 0.5125 - val_loss: 0.8778 - val_accuracy: 0.5500
Epoch 5/8
7/7 [==============================] - 3s 316ms/step - loss: 1.7194 - accuracy: 0.5500 - val_loss: 0.9717 - val_accuracy: 0.4500
Epoch 6/8
7/7 [==============================] - 3s 315ms/step - loss: 1.7105 - accuracy: 0.5375 - val_loss: 1.2379 - val_accuracy: 0.4250
Epoch 7/8
7/7 [==============================] - 3s 317ms/step - loss: 1.3054 - accuracy: 0.6250 - val_loss: 0.7920 - val_accuracy: 0.4750
Epoch 8/8
7/7 [==============================] - 3s 317ms/step - loss: 1.2452 - accuracy: 0.5688 - val_loss: 0.7280 - val_accuracy: 0.6250
```

Ok, so after running the sequential model using the Adamax optimizer and 8 epochs, the accuracy isn't good, but also isnt completely terrible at 0.6250. This means that we didnt overfit the data (as could be seen with a really high accuracy score of like 99.

Try a different architectures like CNN, etc and evaluate on the test data