ashlesha hadke_UEC2021311 2/4/24

```
import os #acess directory
import numpy as np # for complex computation of array  and matrices
import keras # definin g and training model
from keras import layers # layers by layer creation of model
import matplotlib.pyplot as plt #visulization
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,BatchNormalization,Dropout
```

```
!curl -O https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_5340.zip
!unzip -q kagglecatsanddogs_5340.zip
!ls   #List of directory
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  786M  100  786M    0     0  79.0M      0  0:00:09  0:00:09 --:--:-- 74.2M
 CDLA-Permissive-2.0.pdf   kagglecatsanddogs_5340.zip   PetImages  'readme[1].txt'   sample_data
```

```
!ls PetImages     #Now we have a PetImages folder which contain two subfolders, Cat and Dog. Each subfolder contains image files for each ca
```

```
Cat  Dog
```

```
#filter out badly-encoded images that do not feature the string "JFIF" in their header
#JFIF is jpeg FIle Interchange Format
num_skipped = 0 # variable assigned 0

for folder_name in ("Cat", "Dog"): #Loops through each folder "Cat" and "Dog" inside the "PetImages" directory
    folder_path = os.path.join("PetImages", folder_name)

    for fname in os.listdir(folder_path):#Loops through each file in the current folder and constructs the full file path.
        fpath = os.path.join(folder_path, fname)

        try:#Opens the file in binary mode, reads the first 10 bytes, and checks if "JFIF" is present in those bytes.
            fobj = open(fpath, "rb")
            is_jfif = b"JFIF" in fobj.peek(10)
        finally:
            fobj.close()

        if not is_jfif: #If the file is not in JFIF format, increments the counter for skipped images and deletes the file.
            num_skipped += 1
            # Delete corrupted image
            os.remove(fpath)

print(f"Deleted {num_skipped} images.")
```

```
#Generate dataset
image_size = (225, 225) #Specifies the size of the images in the dataset. All images will be resized to this size.
batch_size = 128

train_ds, val_ds = keras.utils.image_dataset_from_directory(
    "PetImages",
    validation_split=0.2,
    subset="both", #pecifies that both the training and validation sets will be returned.
    seed=1337,                          #seed used to control randomness. There is a formula to calculate random number, seed is used in tha
    image_size=image_size,
    batch_size=batch_size,
)
```

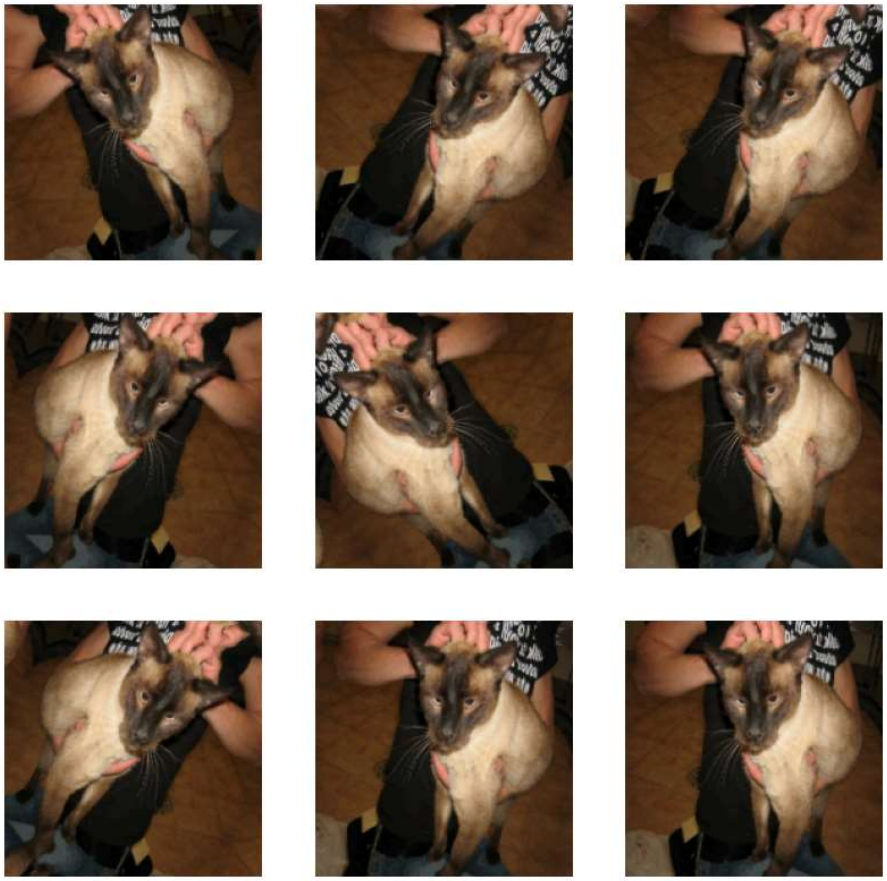
```
Found 23410 files belonging to 2 classes.
Using 18728 files for training.
Using 4682 files for validation.
```

```
#to avoid overfitting used data augmentation , used for image classification
data_augmentation_layers = [

    layers.RandomFlip("horizontal"),# Randomly flips images horizontally.
    layers.RandomRotation(0.1),# Randomly rotates images by a maximum of 0.1 radians
]

def data_augmentation(images):#that applies each data augmentation layer in data_augmentation_layers to the input images sequentially
    for layer in data_augmentation_layers:
        images = layer(images)
    return images


#VISUALIZE THE AUGMENTED DATA
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(np.array(augmented_images[0]).astype("uint8"))
        plt.axis("off")
```



```
# create CNN model

model = Sequential() # used sequential for creating model cnn (layer by layer)

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3))) # added conmvolution layer with neuirns 32.
model.add(BatchNormalization()) # to comprise
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid')) #to reduce dimension

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='relu'))
```

```
model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten()) # convert 3d to 1d

model.add(Dense(128,activation='relu'))# fully connected layer
model.add(Dropout(0.1)) # to avoid overfitting , ignore the weights
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 254, 254, 32)      896

 batch_normalization_16 (Ba  (None, 254, 254, 32)      128
 tchNormalization)

 max_pooling2d_6 (MaxPoolin  (None, 127, 127, 32)      0
 g2D)

 conv2d_9 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_17 (Ba  (None, 125, 125, 64)      256
 tchNormalization)

 max_pooling2d_7 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 conv2d_10 (Conv2D)          (None, 60, 60, 128)       73856

 batch_normalization_18 (Ba  (None, 60, 60, 128)       512
 tchNormalization)

 max_pooling2d_8 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 flatten (Flatten)           (None, 115200)            0

 dense_2 (Dense)             (None, 128)               14745728

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dropout_3 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 1)                 65

=================================================================
Total params: 14848193 (56.64 MB)
Trainable params: 14847745 (56.64 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

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


history = model.fit(train_ds,epochs=10,validation_data=val_ds)
```

```
Epoch 1/10
625/625 [==============================] - 71s 112ms/step - loss: 1.2870 - accuracy: 0.6077 - val_loss: 0.6443 - val_accuracy: 0.6386
Epoch 2/10
625/625 [==============================] - 70s 112ms/step - loss: 0.5226 - accuracy: 0.7400 - val_loss: 0.5353 - val_accuracy: 0.7442
Epoch 3/10
625/625 [==============================] - 71s 113ms/step - loss: 0.4516 - accuracy: 0.7917 - val_loss: 0.4704 - val_accuracy: 0.7834
Epoch 4/10
625/625 [==============================] - 70s 111ms/step - loss: 0.3935 - accuracy: 0.8224 - val_loss: 0.7385 - val_accuracy: 0.6624
Epoch 5/10
625/625 [==============================] - 70s 111ms/step - loss: 0.3376 - accuracy: 0.8543 - val_loss: 0.5054 - val_accuracy: 0.7704
Epoch 6/10
625/625 [==============================] - 70s 112ms/step - loss: 0.2517 - accuracy: 0.8946 - val_loss: 0.8225 - val_accuracy: 0.7428
Epoch 7/10
625/625 [==============================] - 70s 112ms/step - loss: 0.1735 - accuracy: 0.9294 - val_loss: 0.5472 - val_accuracy: 0.7982
Epoch 8/10
625/625 [==============================] - 70s 111ms/step - loss: 0.1200 - accuracy: 0.9551 - val_loss: 0.6331 - val_accuracy: 0.8042
Epoch 9/10
```

```
625/625 [==============================] - 70s 111ms/step - loss: 0.0917 - accuracy: 0.9671 - val_loss: 0.6862 - val_accuracy: 0.8010
Epoch 10/10
625/625 [==============================] - 70s 111ms/step - loss: 0.0755 - accuracy: 0.9744 - val_loss: 0.7166 - val_accuracy: 0.8080
```

```
#data augmentation and dropout are inactive at inference time
import tensorflow as tf
img = keras.utils.load_img("PetImages/Cat/6779.jpg", target_size=image_size)
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
score = float(tf.sigmoid(predictions[0][0]))
print(f"This image is {100 * (1 - score):.2f}% cat and {100 * score:.2f}% dog.")
```

```
1/1 [==============================] - 0s 94ms/step
This image is 99.76% cat and 0.24% dog.
```