```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
```

Image Classification Homework

Author: Abed Ahmed (ASA190005) and Dylan Kapustka (dlk190000)

Description: This homework is about using deep learning to classify images into different categories. We will use a dataset of images and train a model to predict the classes of the images. We will then evaluate the performance of the model and explore techniques for improving the accuracy of the model.

We will be using the "Cats and Dogs" dataset from Kaggle, and will attempt to classify images as on or the other.

```
DATADIR = "/content/drive/MyDrive/PetImages"
CATEGORIES = ['cat', 'dog']
```

```
for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img))
        plt.imshow(img_array)
        plt.show()
        break
    break
```

```
print(img_array)
```

```
     [[[114 107  98]
       [112 105  96]
       [121 114 105]
       ...
       [ 38  39  29]
       [ 42  41  31]
       [ 44  43  33]]

      [[124 117 108]
       [122 115 106]
       [131 124 115]
       ...
       [ 37  36  26]
       [ 37  36  26]
       [ 38  37  27]]

      [[126 119 110]
       [126 119 110]
       [132 125 116]
       ...
       [ 22  20  10]
       [ 21  19   9]
       [ 20  18   8]]
```

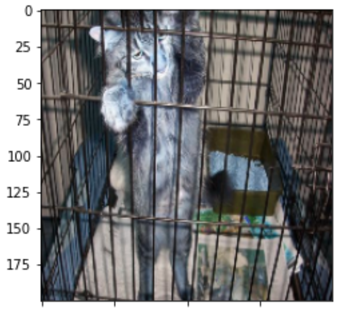
```
...

[[ 59  76  97]
 [ 61  77  94]
 [ 59  68  82]
 ...
 [  0   2   9]
 [ 14  19  28]
 [ 32  37  46]]

[[ 62  79 100]
 [ 65  81  98]
 [ 61  70  84]
 ...
 [  0   5  12]
 [ 17  22  31]
 [ 31  36  45]]

[[ 69  86 107]
 [ 63  79  96]
 [ 62  71  85]
 ...
 [  4  10  17]
 [ 33  38  47]
 [ 12  16  27]]]
```

```
print(img_array.shape)
```

```
(377, 500, 3)
```

```
IMG_SIZE = 200
```

```
new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
plt.imshow(new_array)
plt.show()
```



```
training_data = []
def create_training_data():
    for i in range(len(CATEGORIES)):
        category = CATEGORIES[i]
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
          try:
              img_array = cv2.imread(os.path.join(path,img))
              new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
              training_data.append([new_array, i])
          except:
            pass
create_training_data()
```

```
print(len(training_data))
```

```
3997
```

```
import random
random.shuffle(training_data)
```

```
x = []
y = []
```

```
from sklearn.model_selection import train_test_split
```

```
from tensorflow.keras.utils import to_categorical
```

```
X = []
y = []
for features, label in training_data:
    X.append(features)
    y.append(label)
X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
y = np.array(y)
y = to_categorical(y, num_classes=10)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
import tensorflow as tf
```

Sequential Model

```
batch_size = 128
num_classes = 10
epochs = 10
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(200, 200, 3)))

model.add(MaxPooling2D((2, 2)))


model.add(Flatten())

model.add(Dense(10, activation='softmax'))

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

model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size)

model.evaluate(X_test, y_test)
```

```
     Epoch 1/10
     25/25 [==============================] - 70s 3s/step - loss: 611.7998 - accuracy: 0.4855
     Epoch 2/10
     25/25 [==============================] - 66s 3s/step - loss: 25.4295 - accuracy: 0.5956
     Epoch 3/10
     25/25 [==============================] - 81s 3s/step - loss: 2.3340 - accuracy: 0.7357
     Epoch 4/10
     25/25 [==============================] - 66s 3s/step - loss: 0.8877 - accuracy: 0.8339
     Epoch 5/10
     25/25 [==============================] - 69s 3s/step - loss: 0.4065 - accuracy: 0.9062
     Epoch 6/10
     25/25 [==============================] - 69s 3s/step - loss: 0.2971 - accuracy: 0.9343
     Epoch 7/10
     25/25 [==============================] - 67s 3s/step - loss: 0.1564 - accuracy: 0.9565
     Epoch 8/10
     25/25 [==============================] - 67s 3s/step - loss: 0.1000 - accuracy: 0.9722
     Epoch 9/10
     25/25 [==============================] - 69s 3s/step - loss: 0.0737 - accuracy: 0.9812
     Epoch 10/10
     25/25 [==============================] - 67s 3s/step - loss: 0.0557 - accuracy: 0.9865
     25/25 [==============================] - 7s 264ms/step - loss: 3.1430 - accuracy: 0.5700
     [3.1429736614227295, 0.5699999928474426]
```

```
test_loss, sequential_accuracy = model.evaluate(X_test, y_test)

print(f'sequential accuracy: {sequential_accuracy}')

     25/25 [==============================] - 7s 266ms/step - loss: 3.1430 - accuracy: 0.5700
     sequential accuracy: 0.5699999928474426
```

```
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
```

```
inputs = Input(shape=(200, 200, 3))

x = Conv2D(32, (3, 3), activation='relu')(inputs)

x = MaxPooling2D((2, 2))(x)

x = Flatten()(x)

outputs = Dense(10, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)
```

CNN

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

model.fit(X_train, y_train, epochs=10, batch_size=32)

     Epoch 1/10
     100/100 [==============================] - 75s 743ms/step - loss: 151.7171 - accuracy: 0.5549
     Epoch 2/10
     100/100 [==============================] - 71s 707ms/step - loss: 1.1896 - accuracy: 0.7635
     Epoch 3/10
     100/100 [==============================] - 74s 745ms/step - loss: 0.3982 - accuracy: 0.8911
     Epoch 4/10
     100/100 [==============================] - 71s 706ms/step - loss: 0.2223 - accuracy: 0.9462
     Epoch 5/10
     100/100 [==============================] - 70s 703ms/step - loss: 0.1171 - accuracy: 0.9747
     Epoch 6/10
     100/100 [==============================] - 72s 723ms/step - loss: 0.0739 - accuracy: 0.9834
     Epoch 7/10
     100/100 [==============================] - 70s 702ms/step - loss: 0.0525 - accuracy: 0.9909
     Epoch 8/10
     100/100 [==============================] - 70s 701ms/step - loss: 0.0407 - accuracy: 0.9941
     Epoch 9/10
     100/100 [==============================] - 72s 720ms/step - loss: 0.0331 - accuracy: 0.9953
     Epoch 10/10
     100/100 [==============================] - 70s 703ms/step - loss: 0.0273 - accuracy: 0.9959
     <keras.callbacks.History at 0x7f22c5e3fc70>
```

```
# evaluate the model on the test data
_, accuracy = model.evaluate(X_test, y_test, batch_size=32)

# print the accuracy
print('CNN Accuracy:', accuracy)

     25/25 [==============================] - 6s 253ms/step - loss: 2.7779 - accuracy: 0.6087
     CNN Accuracy: 0.6087499856948853
```

RNN

```
from tensorflow.keras.layers import Input, Reshape, LSTM, Dense
from tensorflow.keras.models import Model
```

```
X = np.array(X).reshape(-1, 200, 200, 3, 1)

inputs = Input(shape=(200, 200, 3, 1))

x = Reshape((200, 600))(inputs)

x = LSTM(128)(x)

outputs = Dense(10, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

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

model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
100/100 [==============================] - 51s 484ms/step - loss: 0.8659 - accuracy: 0.4830
Epoch 2/10
100/100 [==============================] - 54s 538ms/step - loss: 0.7138 - accuracy: 0.4958
Epoch 3/10
100/100 [==============================] - 48s 484ms/step - loss: 0.6976 - accuracy: 0.5052
Epoch 4/10
100/100 [==============================] - 48s 481ms/step - loss: 0.6957 - accuracy: 0.5124
Epoch 5/10
100/100 [==============================] - 48s 477ms/step - loss: 0.6951 - accuracy: 0.5052
Epoch 6/10
100/100 [==============================] - 49s 494ms/step - loss: 0.6938 - accuracy: 0.5111
Epoch 7/10
100/100 [==============================] - 48s 478ms/step - loss: 0.6974 - accuracy: 0.5164
Epoch 8/10
100/100 [==============================] - 48s 480ms/step - loss: 0.6946 - accuracy: 0.5089
Epoch 9/10
100/100 [==============================] - 49s 486ms/step - loss: 0.6921 - accuracy: 0.5124
Epoch 10/10
100/100 [==============================] - 51s 506ms/step - loss: 0.6928 - accuracy: 0.5217
<keras.callbacks.History at 0x7f22c7527ca0>
```

```
_, accuracy = model.evaluate(X_test, y_test, batch_size=32)

print('RNN Accuracy:', accuracy)
```

```
25/25 [==============================] - 5s 197ms/step - loss: 0.7020 - accuracy: 0.4963
RNN Accuracy: 0.4962500035762787
```

Pretrained Model and Transfer learning

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model

inputs = Input(shape=(200, 200, 3))

vgg16 = VGG16(weights='imagenet', include_top=False)

features = vgg16(inputs)

x = Flatten()(features)

outputs = Dense(10, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

for layer in vgg16.layers:
    layer.trainable = False

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

model.fit(X_train, y_train, epochs=17, batch_size=32)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
58889256/58889256 [==============================] - 0s 0us/step
```

```
Epoch 1/10
100/100 [==============================] - 1288s 13s/step - loss: 1.9478 - accuracy: 0.9356
Epoch 2/10
100/100 [==============================] - 1286s 13s/step - loss: 0.6048 - accuracy: 0.9809
Epoch 3/10
100/100 [==============================] - 1286s 13s/step - loss: 0.1734 - accuracy: 0.9941
Epoch 4/10
100/100 [==============================] - 1285s 13s/step - loss: 0.1493 - accuracy: 0.9925
Epoch 5/10
100/100 [==============================] - 1283s 13s/step - loss: 0.0624 - accuracy: 0.9959
Epoch 6/10
100/100 [==============================] - 1287s 13s/step - loss: 0.0819 - accuracy: 0.9953
Epoch 7/10
100/100 [==============================] - 1284s 13s/step - loss: 0.0323 - accuracy: 0.9987
Epoch 8/10
100/100 [==============================] - 1284s 13s/step - loss: 0.0105 - accuracy: 0.9991
Epoch 9/10
100/100 [==============================] - 1284s 13s/step - loss: 0.0651 - accuracy: 0.9978
Epoch 10/10
100/100 [==============================] - 1279s 13s/step - loss: 0.0386 - accuracy: 0.9978
<keras.callbacks.History at 0x7f22c7006790>
```

```
_, accuracy = model.evaluate(X_test, y_test, batch_size=32)

print('Transfer Learning Accuracy:', accuracy)
```

```
25/25 [==============================] - 324s 13s/step - loss: 1.2810 - accuracy: 0.9712
Transfer Learning Accuracy: 0.9712499976158142
```

Overall, It seems that all the models we created did not perform very well accuracy wise. This is likely due to the fact that we had ran into many many problems trying to upload the complete dataset, but if failed everytime. So we had to trim it down, which is likely what caused the lacking accuracy.

Either way,as far the analysys goes. Python seems to be quite a powerful language when it comes to all things ML, but I am curious how much easier R wouold have made this.

As for the Models, in terms of accuracy, they rank:

1 - The pre-trained model & transfer learning

2 - CNN

3 - Plain Sequential Model

4 - RNN

✓ 6m 24s completed at 5:47 PM