Dog, Cat Image Classification

Step 1: Download dataset

1 - For this first you need to downlaod the Kaggel API (kaggle.json file) from your kaggle account. Then put that Kaggle.jason file in .kaggle folder in you PC. After this step run the folowing code to download dataset directly from kaggle.

2 - Make sure to install kaggle libray [pip install kaggle]

In [ ]:
```
#download dataset from kaggle, using Kaggle API

!kaggle datasets download -d salader/dogs-vs-cats
```

Step 2: Unzipe the dataset

In [ ]:
```
#Unzip the downlaoded dataset
import zipfile
zip_ref = zipfile.ZipFile('dogs-vs-cats.zip', 'r')
zip_ref.extractall()
zip_ref.close()
```

Step 3: Import libraries

In [ ]:
```
#Import tensorflow libraries for creating model and classification (CNN Model)
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,BatchNormalization,Dr
```

Step 4: Data Preprocessing

1 - Assign lables, sample from data, set image size

In [ ]:
```
# generators
train_ds = keras.utils.image_dataset_from_directory(
    directory = 'train',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)

validation_ds = keras.utils.image_dataset_from_directory(
    directory = 'test',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)
```

```
Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.
```

2 - Normalize the data values in betweeen 0 & 1 [0 - 1]

In [ ]:
```
# Normalize
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
    return image,label

train_ds = train_ds.map(process)
validation_ds = validation_ds.map(process)
```

Step 5: Model training

In [ ]:
```
# create CNN model

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_sh
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

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(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))
```

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

In [ ]:
```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```

In [ ]:
```
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```

In [ ]:
```
plt.plot(history.history['loss'],color='red',label='train')
plt.plot(history.history['val_loss'],color='blue',label='validation')
```

```
plt.legend()
plt.show()
```

In [ ]:
```
plt.plot(history.history['loss'],color='red',label='train')
plt.plot(history.history['val_loss'],color='blue',label='validation')
plt.legend()
plt.show()
```