```
#Project by "Utkarsh Saxena"
#Image Classification of Cats and Dogs using CNN
```

```
!pip install kaggle opendatasets --upgrade
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.12/dist-packages (1.7.4.5)
Requirement already satisfied: opendatasets in /usr/local/lib/python3.12/dist-packages (0.1.22)
Requirement already satisfied: bleach in /usr/local/lib/python3.12/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2025.8.3)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/dist-packages (from kaggle) (3.4.3)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages (from kaggle) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.12/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.32.4)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.12/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.12/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.12/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.5.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packages (from kaggle) (0.5.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from opendatasets) (8.2.1)
```

```
import json

# Save your credentials into kaggle.json
kaggle_token = {"username":"utkarshsaxena26","key":"b367af0aa3456559b90f5c5a606e8ddb"}

with open("kaggle.json", "w") as f:
    json.dump(kaggle_token, f)
```

```
import opendatasets as od
od.download("https://www.kaggle.com/datasets/salader/dogsvscats")
```

```
Skipping, found downloaded files in "./dogsvscats" (use force=True to force download)
```

```
#Import lib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#Deep learning
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout,BatchNormalization
```

```
train = "/content/dogsvscats/catsvsdogs/train"
test = "/content/dogsvscats/catsvsdogs/test"
```

```
#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.
```

```
#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)
```

```
# Creating CNN Model

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
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',input_shape=(256,256,3)))
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',input_shape=(256,256,3)))
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'))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (BatchNormalization)	(None, 254, 254, 32)	128
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 125, 125, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 60, 60, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 128)	14,745,728
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 14,848,193 (56.64 MB)
Trainable params: 14,847,745 (56.64 MB)

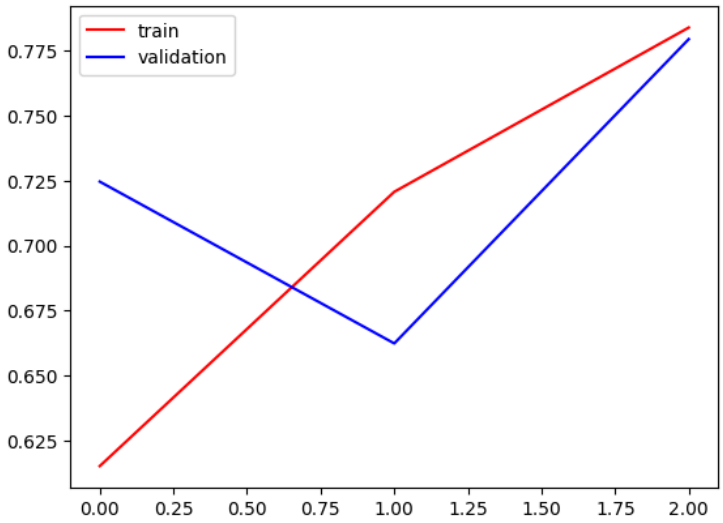
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history = model.fit(train_ds,epochs=3,validation_data = validation_ds)
```

```
Epoch 1/3
625/625 ━━━━━━━━━━━━━━━━━━━━ 70s 91ms/step - accuracy: 0.5822 - loss: 2.5099 - val_accuracy: 0.7246 - val_loss: 0.5675
Epoch 2/3
625/625 ━━━━━━━━━━━━━━━━━━━━ 72s 92ms/step - accuracy: 0.7058 - loss: 0.5715 - val_accuracy: 0.6624 - val_loss: 0.5764
Epoch 3/3
625/625 ━━━━━━━━━━━━━━━━━━━━ 58s 92ms/step - accuracy: 0.7725 - loss: 0.4832 - val_accuracy: 0.7794 - val_loss: 0.5112
```

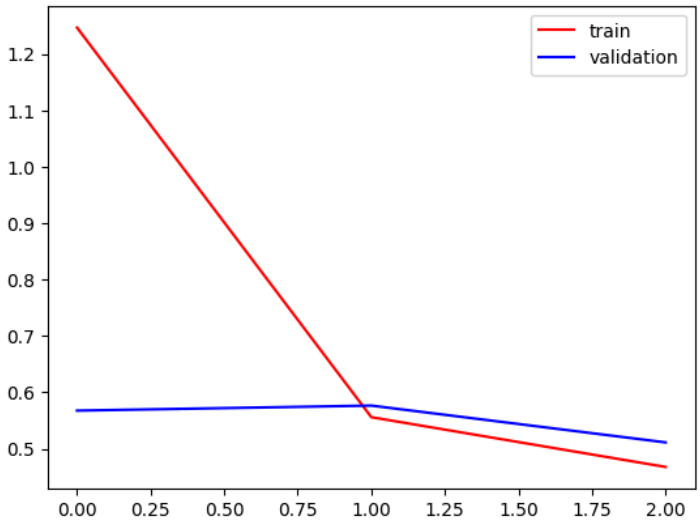
```
#Plotting Accuracy

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




```
#Plotting Loss

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

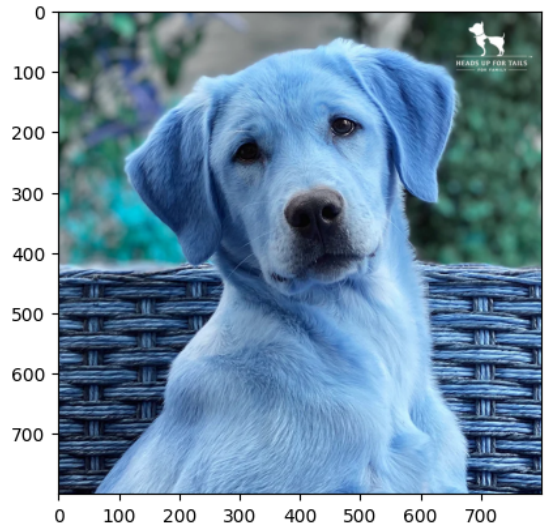



```
import cv2
# Predicted Class- 1 for Dog & 0 for Cat
```

```
test_img = cv2.imread('/content/dog.webp')
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7f8fa43c5490>
```




```
test_img.shape
```

```
(800, 800, 3)
```

```
test_img = cv2.resize(test_img,(256,256))
```

```
test_input = test_img.reshape((1,256,256,3))
```

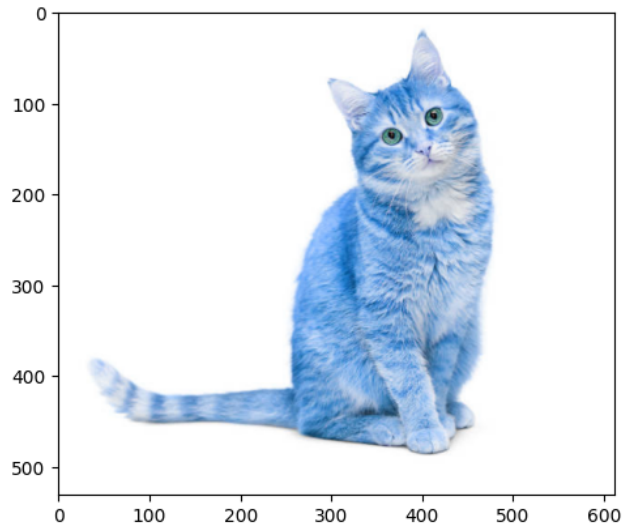
```
model.predict(test_input)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
array([[0.]], dtype=float32)
```

```
test_img2 = cv2.imread('/content/cat.jpg')
```

```
plt.imshow(test_img2)
```

```
<matplotlib.image.AxesImage at 0x7f8fa4478b30>
```




```
test_img2.shape
```

```
(531, 612, 3)
```

```
test_img2 = cv2.resize(test_img2,(256,256))
```

```
test_input2 = test_img2.reshape((1,256,256,3))
```

```
model.predict(test_input2)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
array([[0.]], dtype=float32)
```

```
from google.colab import files
from IPython.display import display, HTML, clear_output
import ipywidgets as widgets
import numpy as np
import matplotlib.pyplot as plt
import io
from PIL import Image as PILImage
import tensorflow as tf

# Custom CSS Styling
display(HTML('''
<style>
  .title { font-family: 'Segoe UI', sans-serif; color: #0F766E; font-size: 32px; font-weight: bold; }
  .subtitle { color: #555; font-size: 16px; font-family: 'Segoe UI'; margin-bottom: 20px; }
  .card {
    background: #F0FDFA;
    border-left: 5px solid #14B8A6;
    padding: 20px;
    margin-top: 20px;
    border-radius: 12px;
    font-family: 'Segoe UI';
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  }
  .confidence-bar {
    height: 20px;
    background: #E0F2F1;
    border-radius: 10px;
    margin-top: 10px;
    overflow: hidden;
  }
  .confidence-fill {
    height: 100%;
    background: #10B981;
    text-align: right;
    color: white;
    padding-right: 8px;
    line-height: 20px;
    font-size: 14px;
    font-weight: bold;
    border-radius: 10px;
  }
</style>
'''))

# Title
display(HTML("<div class='title'>🐾 AI Dog vs Cat Classifier</div>"))
display(HTML("<div class='subtitle'>Upload a photo, click Predict, and see the AI guess with confidence!</div>"))

# Upload & Predict Widgets
upload_widget = widgets.FileUpload(accept='image/*', multiple=False)
predict_btn = widgets.Button(description="🚀 Predict Now", button_style='success')
output_box = widgets.Output()

# UI Layout
ui = widgets.VBox([
    widgets.HTML("<b style='font-size:16px;'>📁 Step 1:</b> Upload an Image"),
    upload_widget,
    widgets.HTML("<b style='font-size:16px;'>👾 Step 2:</b> Click Predict"),
    predict_btn,
    output_box
])
display(ui)

# Predict function
def on_predict_clicked(b):
    with output_box:
        clear_output()
        if not upload_widget.value:
            print("⚠️ Please upload an image.")
            return

        uploaded_file = list(upload_widget.value.values())[0]
        content = uploaded_file['content']
        img_pil = PILImage.open(io.BytesIO(content)).convert('RGB')
        img_resized = img_pil.resize((256, 256))
        img_array = np.array(img_resized)

        # Show uploaded image
        plt.imshow(img_array)
        plt.axis('off')
        plt.title("Uploaded Image")
        plt.show()

        # Prepare for prediction
```

```
        input_img = img_array.reshape((1, 256, 256, 3)).astype('float32') / 255.0
        pred = model.predict(input_img)[0][0]
        label = "🐶 Dog" if pred >= 0.5 else "🐱 Cat"
        confidence = pred if pred >= 0.5 else 1 - pred
        conf_percent = int(confidence * 100)

        # Display results in card
        display(HTML(f'''
            <div class='card'>
                <h3>Prediction: <span style="color: #10B981;">{label}</span></h3>
                <div class="confidence-bar">
                    <div class="confidence-fill" style="width:{conf_percent}%;">{conf_percent}%</div>
                </div>
            </div>
        '''))

# Bind button
predict_btn.on_click(on_predict_clicked)
```

🐾 AI Dog vs Cat Classifier

Upload a photo, click Predict, and see the AI guess with confidence!

📁 **Step 1:** Upload an Image

Upload (1)

🤖 **Step 2:** Click Predict

🚀 Predict Now

Uploaded Image

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
```

Prediction: 🐶 Dog

99%

```
model.save("cat_dog_model.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is
```