Implementation of CNN
on
Cat-Dog classification

A Project Report
in partial fulfillment of the degree

Bachelor of Technology
in
Computer Science & Artificial Intelligence
By

Roll No	Name of the Student
2203A52057	T. Navya Lohitha

School of Computer Science & Artificial Intelligence
SR University, Ananthsagar, Hasanparthy (M), Warangal,
Telangana 506371, India

2024-25

DATASET DESCRIPTION

- **Dataset Name:** Dogs vs. Cats
- **Source:** Kaggle
- **Content:** The dataset consists of images of dogs and cats, where each image belongs to one of the two classes (Dog or Cat). These images are labeled as either "cat" or "dog" based on the animal in the image. The dataset has a total of:

  25,000 training images

  12,500 test images
- **Image Size**: The images are in JPEG format and have varying dimensions. Most of them are resized to fit a neural network's input requirements.

Objective:

The goal of this dataset is to build a model that can predict whether an image contains a cat or a dog. The binary classification task can be tackled using deep learning models such as CNNs (Convolutional Neural Networks), which are well-suited for image classification tasks.

Steps to Approach the Problem:

1. Data Preprocessing:
   - Resize the images to a consistent size, such as 150x150 or 224x224 pixels.
   - Normalize the pixel values to the range [0, 1] by dividing by 255.
   - Split the dataset into training, validation, and test sets (if not already split).
2. Model Building:
   - Use a Convolutional Neural Network (CNN), which is the most commonly used deep learning model for image classification tasks.
   - The architecture can consist of several convolutional layers followed by pooling layers, and eventually fully connected layers.
3. Training:
   - Use binary cross-entropy as the loss function, as it's a binary classification problem.
   - Use accuracy as the evaluation metric.
   - Implement data augmentation to make the model more robust and prevent overfitting.
4. Evaluation:
   - Evaluate the model on the test set.
   - Analyze metrics like accuracy, precision, recall, and F1-score.

Implementation:

Code was implemnted in Kaggle Environement

1. Loading Libraries

```
# Basic
import os
from os import makedirs, listdir
from shutil import copyfile
from random import seed, random
import numpy as np
import pandas as pd

# Visuals
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
from PIL import Image

# Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

# Tensorflow
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, MaxPooling2D, Dropout, Flatten, BatchNormalization,
Conv2D
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

2. Data Extraction and Analysis

```
import os
cat = os.listdir('/kaggle/input/PetImages/Cat')
dog = os.listdir('/kaggle/input/PetImages/Dog')
print("Cat:", cat[:5])
print("Dog:", dog[:5])
```

count images

```
print(len(cat))
print(len(dog))
```

Create lables

```
cat_labels = [1] * 12499
dog_labels = [0] * 12499
labels = cat_labels + dog_labels
print(len(labels))
```

Display Sample Images

```
img = mpimg.imread('/kaggle/input/dog-and-cat-classification-dataset/PetImages/Cat/0.jpg')
plt.imshow(img)
plt.show()

img = mpimg.imread('/kaggle/input/dog-and-cat-classification-dataset/PetImages/Dog/0.jpg')
plt.imshow(img)
plt.show()
```

3. Data Preprocessing

Convert Images to Numpy Arrays

```
from PIL import Image
import numpy as np
data = []
# Cat Images
cat_path = '/kaggle/input/dog-and-cat-classification-dataset/PetImages/Cat/'
for img_file in cat:
    image = Image.open(cat_path + img_file).resize((128, 128)).convert('RGB')
    data.append(np.array(image))
# Dog Images
dog_path = '/kaggle/input/dog-and-cat-classification-dataset/PetImages/Dog/'
for img_file in dog:
    image = Image.open(dog_path + img_file).resize((128, 128)).convert('RGB')
    data.append(np.array(image))
x = np.array(data)
y = np.array(labels)
```

Train-Test Split and Scaling data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
x_train_scal = x_train / 255
x_test_scal = x_test / 255
```

4. Model Development

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='sigmoid'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc'])
model.fit(x_train_scal, y_train, validation_split=0.1, epochs=30)

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

5. Prediction

```
input_image_path = input('Path of the image to be predicted: ')
input_image = cv2.imread(input_image_path)
cv2_imshow(input_image)
input_image_resized = cv2.resize(input_image, (128, 128))
input_image_scaled = input_image_resized / 255
input_image_reshaped = np.reshape(input_image_scaled, [1, 128, 128, 3])
input_prediction = model.predict(input_image_reshaped)
input_pred_label = np.argmax(input_prediction)
if input_pred_label == 1:
  print('The person in the image is wearing a mask')
else:
  print('The person in the image is not wearing a mask')
```

RESULTS

Loading images

```
import os

cat = os.listdir('/kaggle/input/PetImages/Cat')
dog = os.listdir('/kaggle/input/PetImages/Dog')

print("Cat:", cat[:5])
print("Dog:", dog[:5])
```

```
Cat: ['7981.jpg', '6234.jpg', '1269.jpg', '3863.jpg', '6241.jpg']
Dog: ['7981.jpg', '6234.jpg', '1269.jpg', '3863.jpg', '6241.jpg']
```

Data Extraction and Analysis

```
print(len(cat))
print(len(dog))
```

```
12499
12499
```

```
cat_labels = [1]*12499
dog_labels = [0]*12499
```

```
print(cat_labels[:5])
print(dog_labels[:5])
```

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
labels =cat_labels + dog_labels
print(len(labels))
```

```
24998
```

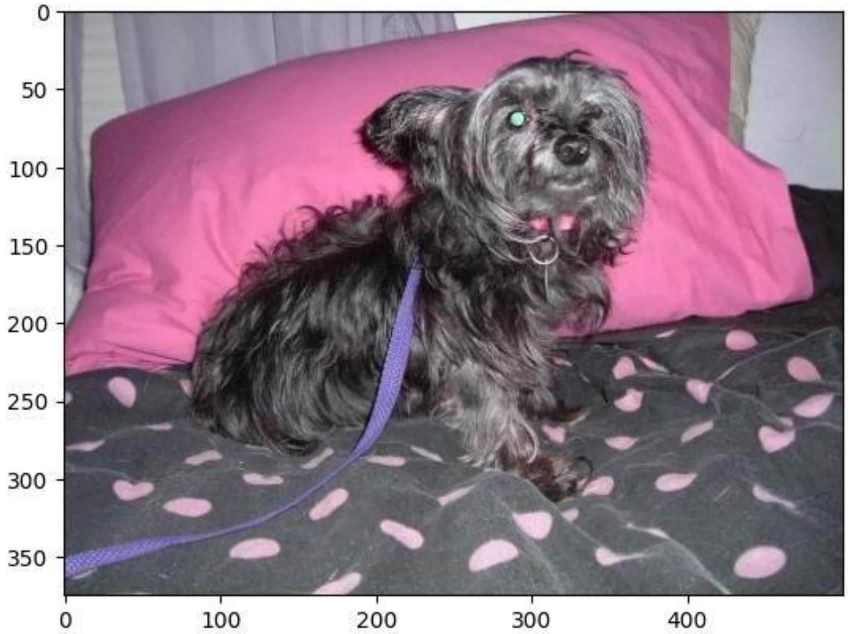
```
print(labels[:5])
print(labels[-5:])
```

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

\

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg # import the module

img = mpimg.imread('/kaggle/input/PetImages/Cat/0.jpg') # Now mpimg is defined and can be used
imgplot = plt.imshow(img)
plt.show()
```

```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg # import the module

img = mpimg.imread('/kaggle/input/PetImages/Dog/0.jpg') # Now mpimg is defined and can be used
imgplot = plt.imshow(img)
plt.show()
```




```python
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 128)	7,372,928
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 2)	130

Total params: 7,400,706 (28.23 MB)

Trainable params: 7,400,706 (28.23 MB)

Non-trainable params: 0 (0.00 B)

MODEL TRAINING

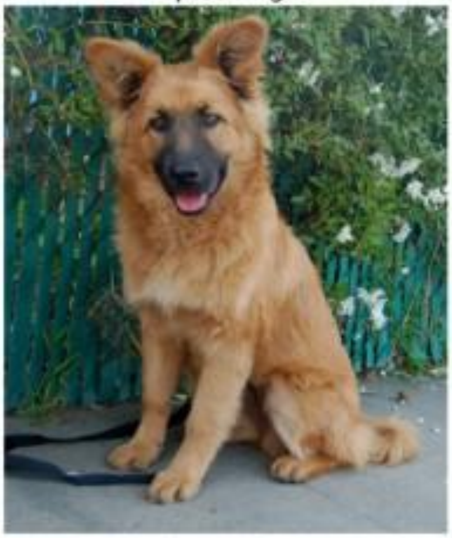
```
Epoch 1/30
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super().__init_
they will be ignored.
  self._warn_if_super_not_called()
563/563 ━━━━━━━━━━━━━━━━━━━━ 79s 132ms/step - accuracy: 0.6884 - loss: 0.6158 - val_accuracy: 0.7379 - val_loss: 0.5728 - learning_rate: 0.0010
Epoch 2/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 123ms/step - accuracy: 0.7178 - loss: 0.5840 - val_accuracy: 0.7384 - val_loss: 0.5533 - learning_rate: 0.0010
Epoch 3/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 123ms/step - accuracy: 0.7276 - loss: 0.5658 - val_accuracy: 0.7129 - val_loss: 0.5867 - learning_rate: 0.0010
Epoch 4/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 123ms/step - accuracy: 0.7368 - loss: 0.5483 - val_accuracy: 0.7409 - val_loss: 0.5253 - learning_rate: 0.0010
Epoch 5/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 75s 131ms/step - accuracy: 0.7474 - loss: 0.5300 - val_accuracy: 0.7394 - val_loss: 0.5252 - learning_rate: 0.0010
Epoch 6/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 75s 132ms/step - accuracy: 0.7524 - loss: 0.5174 - val_accuracy: 0.7574 - val_loss: 0.5054 - learning_rate: 0.0010
Epoch 7/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 124ms/step - accuracy: 0.7525 - loss: 0.5132 - val_accuracy: 0.7674 - val_loss: 0.4908 - learning_rate: 0.0010
Epoch 8/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 75s 131ms/step - accuracy: 0.7569 - loss: 0.5085 - val_accuracy: 0.7509 - val_loss: 0.5073 - learning_rate: 0.0010
Epoch 9/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 122ms/step - accuracy: 0.7624 - loss: 0.4996 - val_accuracy: 0.7869 - val_loss: 0.4758 - learning_rate: 0.0010
Epoch 10/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 73s 128ms/step - accuracy: 0.7773 - loss: 0.4830 - val_accuracy: 0.7924 - val_loss: 0.4616 - learning_rate: 0.0010
Epoch 11/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 75s 132ms/step - accuracy: 0.7769 - loss: 0.4813 - val_accuracy: 0.7909 - val_loss: 0.4667 - learning_rate: 0.0010
Epoch 12/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 72s 126ms/step - accuracy: 0.7817 - loss: 0.4694 - val_accuracy: 0.7864 - val_loss: 0.4635 - learning_rate: 0.0010
Epoch 13/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 71s 125ms/step - accuracy: 0.7819 - loss: 0.4707 - val_accuracy: 0.7779 - val_loss: 0.4774 - learning_rate: 0.0010
Epoch 14/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 71s 125ms/step - accuracy: 0.7932 - loss: 0.4554 - val_accuracy: 0.8209 - val_loss: 0.4291 - learning_rate: 2.0000e-04
Epoch 15/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 75s 132ms/step - accuracy: 0.8045 - loss: 0.4408 - val_accuracy: 0.8109 - val_loss: 0.4339 - learning_rate: 2.0000e-04
Epoch 16/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 75s 133ms/step - accuracy: 0.8085 - loss: 0.4274 - val_accuracy: 0.8149 - val_loss: 0.4217 - learning_rate: 2.0000e-04
Epoch 17/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 72s 126ms/step - accuracy: 0.8127 - loss: 0.4235 - val_accuracy: 0.8074 - val_loss: 0.4305 - learning_rate: 2.0000e-04
Epoch 18/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 73s 128ms/step - accuracy: 0.8091 - loss: 0.4261 - val_accuracy: 0.8129 - val_loss: 0.4298 - learning_rate: 2.0000e-04
Epoch 19/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 73s 127ms/step - accuracy: 0.8086 - loss: 0.4275 - val_accuracy: 0.8074 - val_loss: 0.4237 - learning_rate: 2.0000e-04
Epoch 20/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 71s 124ms/step - accuracy: 0.8152 - loss: 0.4160 - val_accuracy: 0.8134 - val_loss: 0.4174 - learning_rate: 4.0000e-05
Epoch 21/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 71s 124ms/step - accuracy: 0.8149 - loss: 0.4261 - val_accuracy: 0.8144 - val_loss: 0.4177 - learning_rate: 4.0000e-05
Epoch 22/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 73s 129ms/step - accuracy: 0.8178 - loss: 0.4190 - val_accuracy: 0.8139 - val_loss: 0.4220 - learning_rate: 4.0000e-05
Epoch 23/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 123ms/step - accuracy: 0.8202 - loss: 0.4180 - val_accuracy: 0.8069 - val_loss: 0.4250 - learning_rate: 4.0000e-05
Epoch 24/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 122ms/step - accuracy: 0.8130 - loss: 0.4230 - val_accuracy: 0.8084 - val_loss: 0.4089 - learning_rate: 8.0000e-06
Epoch 25/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 68s 120ms/step - accuracy: 0.8169 - loss: 0.4114 - val_accuracy: 0.8054 - val_loss: 0.4225 - learning_rate: 8.0000e-06
Epoch 26/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 72s 126ms/step - accuracy: 0.8255 - loss: 0.4032 - val_accuracy: 0.8244 - val_loss: 0.4084 - learning_rate: 8.0000e-06
Epoch 27/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 71s 125ms/step - accuracy: 0.8216 - loss: 0.4068 - val_accuracy: 0.8084 - val_loss: 0.4094 - learning_rate: 8.0000e-06
Epoch 28/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 70s 123ms/step - accuracy: 0.8154 - loss: 0.4195 - val_accuracy: 0.8114 - val_loss: 0.4160 - learning_rate: 8.0000e-06
Epoch 29/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 71s 125ms/step - accuracy: 0.8170 - loss: 0.4165 - val_accuracy: 0.8234 - val_loss: 0.4064 - learning_rate: 8.0000e-06
Epoch 30/30
563/563 ━━━━━━━━━━━━━━━━━━━━ 73s 128ms/step - accuracy: 0.8224 - loss: 0.4049 - val_accuracy: 0.8194 - val_loss: 0.4092 - learning_rate: 8.0000e-06
<keras.src.callbacks.history.History at 0x78667c277520>
```

PREDICTION:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model

# Load saved model
model = load_model('model.h5')

# Input image path
input_image_path = input('Path of the image to be predicted: ')

# Read and display
input_image = cv2.imread(input_image_path)
input_image_rgb = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)
plt.imshow(input_image_rgb)
plt.axis('off')
plt.title('Input Image')
plt.show()

# Preprocessing
input_image_resized = cv2.resize(input_image, (128,128))
input_image_scaled = input_image_resized / 255.0
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

# Predict
input_prediction = model.predict(input_image_reshaped)
print("Prediction Probability:", input_prediction[0][0])

# Binary Classification (0 = Dog, 1 = Cat)
input_pred_label = (input_prediction[0][0] > 0.5).astype(int)

if input_pred_label == 1:
    print('The animal in the image is a **Cat**')
else:
    print('The animal in the image is a **Dog**')
```

```
Path of the image to be predicted:  /kaggle/input/dog-and-cat-classification-dataset/PetImages/Dog/10014.jpg
```

Input Image



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 225ms/step
Prediction Probability: 0.09162871
The animal in the image is a **Dog**
```

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model

# Load saved model
model = load_model('model.h5')

# Input image path
input_image_path = input('Path of the image to be predicted: ')

# Read and display
input_image = cv2.imread(input_image_path)
input_image_rgb = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)
plt.imshow(input_image_rgb)
plt.axis('off')
plt.title('Input Image')
plt.show()

# Preprocessing
input_image_resized = cv2.resize(input_image, (128,128))
input_image_scaled = input_image_resized / 255.0
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

# Predict
input_prediction = model.predict(input_image_reshaped)
print("Prediction Probability:", input_prediction[0][0])

# Binary Classification (0 = Dog, 1 = Cat)
input_pred_label = (input_prediction[0][0] > 0.5).astype(int)

if input_pred_label == 1:
    print('The animal in the image is a **Cat**')
else:
    print('The animal in the image is a **Dog**')
```

```
Path of the image to be predicted:  /kaggle/input/dog-and-cat-classification-dataset/PetImages/Cat/100.jpg
```

Input Image

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 222ms/step
Prediction Probability: 0.5203394
The animal in the image is a **Cat**
```