Cat_Vs_Dog_Classification

June 24, 2023

1 Problem Statement: Cat V/s Dog Classification

2 Description:

Cat vs dog classification is a popular computer vision task that involves training a model to distinguish between images of cats and dogs. This task can be challenging, as there is a lot of variation in the appearance of cats and dogs, even within the same breed. However, with the use of deep learning techniques, it is possible to achieve high accuracy in cat vs dog classification.

One of the most common approaches to cat vs dog classification is to use a convolutional neural network (CNN). CNNs are a type of deep learning model that are specifically designed for image classification tasks. They work by learning to identify patterns in images, and can be trained on large datasets of labeled images.

Another approach to cat vs dog classification is to use a transfer learning model. Transfer learning is a technique where a pre-trained model is used as a starting point for training a new model. This can be helpful for cat vs dog classification, as there are many pre-trained CNN models that have been trained on large datasets of images.

The accuracy of cat vs dog classification models can vary depending on the dataset used to train the model, the architecture of the model, and the optimization techniques used. However, it is possible to achieve accuracies of over 95% with well-trained models.

3 1.0 Importing Libraries

[ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import tensorflow_datasets as tfds
from warnings import filterwarnings
filterwarnings("ignore")
```

```
[ ]: from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

4 2.0 Loading Datasets

```
[ ]: datasets,info=tfds.load('cats_vs_dogs',with_info=True,as_supervised=True)
```

```
Downloading and preparing dataset 786.68 MiB (download: 786.68 MiB, generated:
Unknown size, total: 786.68 MiB) to
/root/tensorflow_datasets/cats_vs_dogs/4.0.0…

Dl Completed…: 0 url [00:00, ? url/s]

Dl Size…: 0 MiB [00:00, ? MiB/s]

Generating splits…:   0%|          | 0/1 [00:00<?, ? splits/s]

Generating train examples…:   0%|          | 0/23262 [00:00<?, ? examples/s]

WARNING:absl:1738 images were corrupted and were skipped

Shuffling /root/tensorflow_datasets/cats_vs_dogs/4.0.0.incompleteEKYZ8C/
↪cats_vs_dogs-train.tfrecord*…:   0%|…

Dataset cats_vs_dogs downloaded and prepared to
/root/tensorflow_datasets/cats_vs_dogs/4.0.0. Subsequent calls will reuse this
data.
```

5 Datasets Information:

Cat and dog datasets are a collection of images of cats and dogs that are used to train and evaluate machine learning models for image classification. These datasets are typically divided into two classes: cats and dogs. Each image in the dataset is labeled with its corresponding class.

One of the most popular cat and dog datasets is the Dogs vs. Cats dataset: https://www.kaggle.com/c/dogs-vs-cats, which is hosted on Kaggle. This dataset contains over 25,000 images of cats and dogs, which were manually classified by people at thousands of animal shelters across the United States. The images in this dataset are of varying quality, but they are a good starting point for training and evaluating machine learning models for image classification.

Another popular cat and dog dataset is the Cats vs. Dogs dataset: https://www.tensorflow.org/datasets/catalog/cats_vs_dogs from TensorFlow Datasets. This dataset contains over 10,000 images of cats and dogs, which were collected from the internet. The images in this dataset are of higher quality than the images in the Dogs vs. Cats dataset, but they are also more challenging to classify.

To evaluate the performance of a machine learning model for image classification, a confusion matrix is often used. A confusion matrix is a table that shows the true positives, true negatives, false positives, and false negatives of a classification model. The true positives are the number of images that were correctly classified as cats or dogs. The true negatives are the number of images that were correctly classified as not cats or dogs. The false positives are the number of images that were incorrectly classified as cats or dogs. The false negatives are the number of images that were incorrectly classified as not cats or dogs.

Here We are going to use 1st Datasets

[ ]:
```
class_names=info.features['label'].names
class_names
```

[ ]:
```
['cat', 'dog']
```

- Cat and Dog

[ ]:
```
import os
import tensorflow as tf

# This code will save the images from the `dataset['train']` dataset to the␣
 ↪`cats_vs_dogs/train` directory.
# The images will be saved in subdirectories based on their labels.

class_names = ['cat', 'dog']

for i, example in enumerate(datasets['train']):
  # The `example` variable is a tuple containing the image and label for the␣
 ↪current iteration.

  image, label = example

  # Save the image to the appropriate subdirectory.

  save_dir = './cats_vs_dogs/train/{}'.format(class_names[label])
  os.makedirs(save_dir, exist_ok=True)

  filename = save_dir + "/" + "{}_{}.jpg".format(class_names[label], i)
  tf.keras.preprocessing.image.save_img(filename, image.numpy())

  # Break after saving the first image.

  # break
```

6 3.0 Building CNN Model

[ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen=ImageDataGenerator(rescale=1/255,validation_split=0.2,rotation_range=10,
                           width_shift_range=0.1,height_shift_range=0.1,
                           shear_range=0.1,zoom_range=0.1,
                           horizontal_flip=True)
```

[ ]:
```
train_generator = datagen.flow_from_directory('/content/cats_vs_dogs/train',
                                              target_size = (150, 150),
                                              batch_size=32,
                                              class_mode='binary',
                                              subset='training')
```

```
Found 18611 images belonging to 2 classes.
```

[ ]:
```
validation_generator = datagen.flow_from_directory('/content/cats_vs_dogs/
 ↪train',
                                              target_size = (150, 150),
                                              batch_size=32,
                                              class_mode='binary',
                                              subset='validation')
```

```
Found 4651 images belonging to 2 classes.
```

6.0.1 The Model

[ ]:
```
from tensorflow.keras.layers import Dropout
model=Sequential()

# 1st Layer
model.add(Conv2D(32,kernel_size=3,activation='relu',input_shape=(150,150,3)))
model.add(MaxPooling2D(2))

# 2nd Layer
model.add(Conv2D(64,kernel_size=3,activation='relu'))
model.add(MaxPooling2D(2))

# 3rd Layer
model.add(Conv2D(128,kernel_size=3,activation='relu'))
model.add(MaxPooling2D(2))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

[ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 flatten (Flatten)           (None, 36992)             0

 dropout (Dropout)           (None, 36992)             0

 dense (Dense)               (None, 512)               18940416

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 19,034,177
Trainable params: 19,034,177
Non-trainable params: 0
_________________________________________________________________
```

- Let's Compile The Model

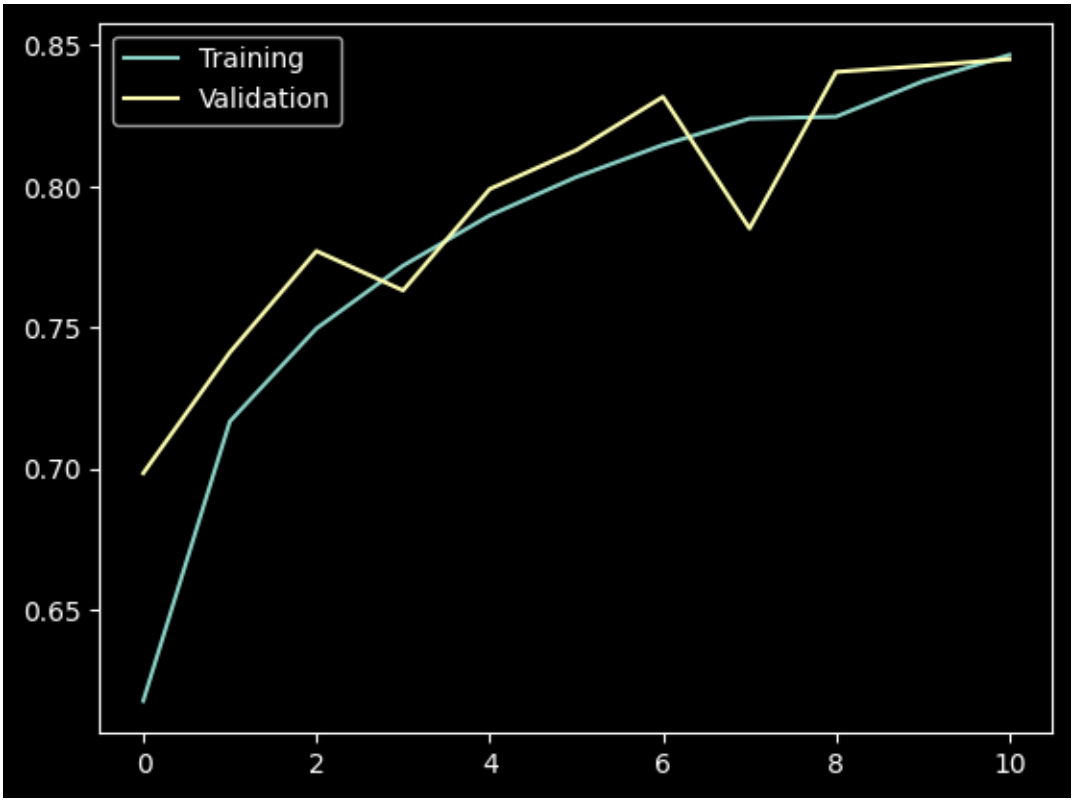
[ ]:
```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

# let's watch how model train
history=model.
  ↪fit(train_generator,epochs=11,validation_data=validation_generator)
```

```
Epoch 1/11
582/582 [==============================] - 197s 318ms/step - loss: 0.6497 -
accuracy: 0.6177 - val_loss: 0.5712 - val_accuracy: 0.6983
Epoch 2/11
582/582 [==============================] - 182s 312ms/step - loss: 0.5509 -
accuracy: 0.7167 - val_loss: 0.5076 - val_accuracy: 0.7411
Epoch 3/11
582/582 [==============================] - 182s 313ms/step - loss: 0.5032 -
```

```
accuracy: 0.7496 - val_loss: 0.4755 - val_accuracy: 0.7770
Epoch 4/11
582/582 [==============================] - 182s 313ms/step - loss: 0.4724 -
accuracy: 0.7719 - val_loss: 0.4826 - val_accuracy: 0.7631
Epoch 5/11
582/582 [==============================] - 182s 312ms/step - loss: 0.4458 -
accuracy: 0.7896 - val_loss: 0.4348 - val_accuracy: 0.7990
Epoch 6/11
582/582 [==============================] - 182s 313ms/step - loss: 0.4279 -
accuracy: 0.8032 - val_loss: 0.4125 - val_accuracy: 0.8127
Epoch 7/11
582/582 [==============================] - 190s 326ms/step - loss: 0.4088 -
accuracy: 0.8146 - val_loss: 0.3775 - val_accuracy: 0.8316
Epoch 8/11
582/582 [==============================] - 183s 315ms/step - loss: 0.3896 -
accuracy: 0.8239 - val_loss: 0.4526 - val_accuracy: 0.7850
Epoch 9/11
582/582 [==============================] - 183s 315ms/step - loss: 0.3837 -
accuracy: 0.8246 - val_loss: 0.3642 - val_accuracy: 0.8405
Epoch 10/11
582/582 [==============================] - 181s 311ms/step - loss: 0.3648 -
accuracy: 0.8371 - val_loss: 0.3607 - val_accuracy: 0.8426
Epoch 11/11
582/582 [==============================] - 180s 310ms/step - loss: 0.3506 -
accuracy: 0.8465 - val_loss: 0.3503 - val_accuracy: 0.8450
```

[ ]:
```python
import matplotlib.pyplot as plt
history.history
plt.style.use('dark_background')
plt.plot(history.history['accuracy'], label='Training')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.legend(['Training', 'Validation'])
```

[ ]:
```
<matplotlib.legend.Legend at 0x7f611c0b1a80>
```



- **Look Training and Validation accuracy almost same, Our model tarined well.:**

[ ]:
```
# save model
model.save('cats_vs_dogs.h5')
```

[ ]:
```
model_load = tf.keras.models.load_model('cats_vs_dogs.h5')
```

[ ]:
```
import numpy as np
import requests
from PIL import Image
from tensorflow.keras.preprocessing import image

img_url = "https://i.natgeofe.com/n/548467d8-c5f1-4551-9f58-6817a8d2c45e/
 ↪NationalGeographic_2572187_square.jpg"
img = Image.open(requests.get(img_url, stream=True).raw).resize((150, 150))

image_array = image.img_to_array(img)

img = np.expand_dims(image_array, axis=0)

img = img/255
```

```
prediction = model.predict(img)

TH = 0.5
prediction = int(prediction[0][0]>TH)
classes = {v:k for k,v in train_generator.class_indices.items()}
classes[prediction]
```

```
1/1 [==============================] - 0s 367ms/step
```

[ ]:
```
'cat'
```

7 Thank You!