DOG BREED CLASSIFICATION BY DEEP LEARNING

ABSTRACT

Dog breed Categorization is essential for many reasons, particularly for understanding individual breeds type identification. A set of a breed images of dogs are used to classify. This species classifying application is planned to be developed using Convolution Neural Networks (CNN) to achieve maximum accuracy and CNN is a supervised learning technique which needs both input data and target output data to be supplied. These are classified by using their labels to provide a learned model for future data analysis. The method begins with a transfer learning by retraining existing pre-trained CNN on the dog breed dataset. Then the image augmentation with various settings is also applied on the training dataset, to improve the classification performance. The images are converted to a single label of dimension with image processing. This analysis to shorten the most similar features into one group to make an easy study of the features into the deep neural networks and the facial features are stored in a vector form. This prepared vector will be compared with each feature of the dog into the dataset and will give greater accuracy.

PROBLEM STATEMENT

The problem statement for dog breed classification using deep learning CNN is to accurately identify the breed of a dog from an input image. This is a challenging computer vision task due to the large variations in appearance among different dog breeds, such as differences in size, shape, color, and texture. The goal is to develop a deep learning model that can generalize well to new images and achieve high accuracy in identifying dog breeds. The problem statement involves acquiring and preprocessing a large dataset of dog images, selecting an appropriate CNN architecture, training the model using backpropagation and an optimization algorithm, evaluating the performance on validation and test sets, and fine-tuning the hyperparameters to achieve better accuracy.

Additionally, the model should be able to handle different types of images, such as images with multiple dogs or images taken from different angles or lighting conditions. The problem statement also involves addressing challenges such as overfitting, underfitting, and class imbalance. Data augmentation techniques can be used to increase the size and diversity of the dataset and prevent overfitting. Transfer learning can be used to leverage pre-trained models and improve the performance of the model with limited training data. Finally, interpretability techniques such as visualization of intermediate feature maps can be used to gain insight into how the model is identifying different dog breeds.

In summary, the problem statement for dog breed classification using deep learning CNN is to develop a model that can accurately identify the breed of a dog from an input image, handle different types of images, and generalize well to new data. This involves acquiring and preprocessing the dataset, selecting an appropriate CNN architecture, training, and evaluating the model, and addressing challenges such as overfitting.

CODE:

```
import os
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2,preprocess_input
from tensorflow.keras.layers import Dense,Conv2D,GlobalAvgPool2D,Input
from tensorflow.keras.preprocessing.image import load_img,ImageDataGenerator
from tensorflow.keras import callbacks,optimizers
import numpy as np
from google.colab import drive

drive.mount('/content/drive')

%cd drive/MyDrive/sem/

!unzip Dog_Dataset.zip

os.listdir("Dog_Dataset")

!ls Dog_Dataset/
for i in os.listdir("Dog_Dataset"):
  print(i,len(os.listdir("Dog_Dataset/"+i)))

try:
   os.mkdir("train")
  os.mkdir("test")
except:
  pass
for i in os.listdir("Dog_Dataset"):
  try:
    os.mkdir("train/"+i)
    os.mkdir("test/"+i)
  except:
    pass
  for j in os.listdir("Dog_Dataset/"+i)[:35]:
    os.rename("Dog_Dataset/"+i+"/"+j,"train/"+i+"/"+j)
  for j in os.listdir("Dog_Dataset/"+i)[:15]:
    os.rename("Dog_Dataset/"+i+"/"+j,"test/"+i+"/"+j)

def img_Data(dir_path,target_size,batch,class_lst,preprocessing,):
  if preprocessing:
    gen_object =ImageDataGenerator(preprocessing_function=preprocessing)
  else:
    gen_object =ImageDataGenerator()

  return(gen_object.flow_from_directory(dir_path,
                                        target_size= target_size,
                                        batch_size=batch,
                                        class_mode='sparse',
                                        classes=class_lst,
                                        shuffle=True))
 train_data_gen=img_Data("train",(224,224),500,os.listdir("train"),preprocess_input)
valid_data_gen =img_Data("test",(224,224),500,os.listdir("test"),preprocess_input)
 base_model=tf.keras.applications.mobilenet_v2.MobileNetV2(
              input_shape=(224,224,3),
              alpha=1.0,
              include_top=False,
              weights='imagenet',
              input_tensor=None,
              pooling=None,
              classes=1000,
              classifier_activation='softmax')
base_model.trainable=False
model=tf.keras.models.Sequential()
model.add(base_model)
model.add(GlobalAvgPool2D())
model.add(Dense(1024,activation='relu'))
model.add(Dense(1024,activation='softmax'))

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
elst=callbacks.EarlyStopping(monitor='val_loss',patience=5,mode='min')
save_ck=callbacks.ModelCheckpoint('.mdl_wt.hdf5',save_best_only=True,monitor='val_loss',
mode='min')
```

```
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

model.fit(train_data_gen,batch_size=500,validation_data=valid_data_gen,callbacks=[elst,save
_ck],epochs=10)
```