Dog breed identification using transfer learning

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	3 MARKS

6.3 - Data Pre-Processing

1: Organizing the Images into Different Classes.

```
[ ]  import os
     import shutil
     import sys

[ ]  dataset_dir = '/content/train'
     labels = pd.read_csv('/content/labels.csv')

[ ]  import os

     def make_dir(x):
         if os.path.exists(x)==False:
             os.makedirs(x)

     base_dir = './subset'
     make_dir(base_dir)

[ ]  train_dir = os.path.join(base_dir, 'train')
     make_dir(train_dir)

[ ]  breeds = labels.breed.unique()
     for breed in breeds:
         # Make folder for each breed
         _ = os.path.join(train_dir, breed)
         make_dir(_)

         # Copy images to the corresponding folders
         images = labels[labels.breed == breed]['id']
         for image in images:
             source = os.path.join(dataset_dir, f'{image}.jpg')
             destination = os.path.join(train_dir, breed,f'{image}.jpg')
             shutil.copyfile(source, destination)
```

The Images should be organized based on the Image id's. So that Training the Model will be simpler. For each image ID (image_id) in the current breed, the source path is formed by combining the dataset_dir and the image filename (f'{image_id}.jpg'). The destination path is formed by combining the breed_folder and the same image filename. shutil. copyfile () is used to copy the image from the source path to the destination path.

2: Import the ImageDataGenerator library.

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

Let us import the ImageDataGenerator class from tensorflow Keras.

```
#import image datagenerator library
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

3: Configure ImageDataGenerator class

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation. There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

```
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

4: Apply ImageDataGenerator functionality to Trainset and Test set:

Let us apply ImageDataGenerator functionality to Train set and Test set by using the following code. For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories affenpinscher, beagle, appenzeller, basset, bluetick, boxer, cairn, doberman, german_shepherd, golden_retriever, kelpie, komondor, leonberg, mexican_hairless, pug, redbone, shih-tzu, toy_poodle, vizsla, whippe, together with labels 0 to 19.

Arguments:

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 32.
- target_size: Size to resize images after they are read from disk.
- class_mode:

  'int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).

  'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).

- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
- None (no labels).

```
[ ]  datagen = ImageDataGenerator()
     generator = datagen.flow_from_directory(
         dataset,
         target_size=(224, 224),  # Adjust target size as needed
         batch_size=32,
         class_mode='categorical',
         shuffle=False,  # Ensure order is maintained for class indices
         classes=selected_classes  # Specify the selected classes
     )

     Found 1683 images belonging to 20 classes.
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
```

We notice that 1683 images belong to 20 classes for training purposes.