DOG BREED

IDENTIFICATION

18CSC305J – Artificial Intelligence

(2018 Regulation)

III Year / VI Semester

Academic Year: 2022 -2023

BY

Vishal Agarwal [RA2011026010063]

Rayansh Srivastava [RA2011026010069]

Himanshu [RA20111026010084]

FACULTY OF ENGINEERING AND TECHNOLOGYSCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
Kattankulathur, Kancheepuram

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
COLLEGE OF ENGINEERING & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report**"Dog Breed Identification Sytem"** is the bonafide work of **" Vishal Agarwal(RA2011026010063), Rayansh Srivastava(RA2011026010069), Himanshu(RA2011026010084)"** of III Year/VI Sem B.tech(CSE) who carried out the mini project work under my supervision for the course 18CSC305J- Artificial Intelligence in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

Signature	**Signature**
Dr. M.S. Abirami	Dr R. Annie Uthra
Associate Professor	HoD and Professor
Department of	Department of
Computational Intelligence	Computational Intelligence

Introduction

This project helps to identify dog breeds from images. This is a fine-grained classification problem: all breeds of Canis lupus familiar will share similar body features and over-all structure, so differences between breeds is a difficult problem. Moreover, there is low inter-breed and high intrabreed variation; in other words, there are few differences between breeds and large differences within breeds, difference in size, shape, and color. However, dogs are both the most morphologically and genetically diverse species on Earth. It is difficult to identify breeds because of diversity are compounded by the stylistic differences of photographs used in dataset, which features dogs of the same breed in a variation of lightings and positions.

If the image of a dog is provided then the algorithm will work for finding the breed of dog and features similarity in the breed and if the human image is supplied it determines the facial features available in a dog of human and vice-versa. The images of human beings and dogs both are considered for the breed classification to find available percentage of features in humans of dogs and vice-versa. This research paperwork has used the principal factor analysis to shorten in the most similar features into one group to make an easy study of the features into the deep neural networks.

To overcome this valuable time, we use Transfer Learning. In computer vision, transfer learning refers to use of a pretrained models to train the Convolutional Neural Network. By Transfer learning, a pre-trained model is trained toprovide solution to classification problem which is like the classification problem we have. In this project we are using pretrained model like VGG16.

One of the areas where deep learning is used extremely is image classification. Convolutional Neural Networks is the constantly used deep learning method for image classification. CNN is parallel to artificial neural networks which have learnable weights and biases. The difference between ANN and CNN is that in CNN's filters is process over the whole image and are the effective methods for image recognition and classification problems

Problem Statement

The identification of dog breeds is a challenging task due to the high similarity between certain breeds. There are over 300 recognized dog breeds in the world, each with its own unique physical and behavioral characteristics. Accurately identifying a dog’s breed can be important for various reasons such as providing appropriate care, training, and nutrition.

The goal of this project is to develop a system that can accurately classify images of dogs into their respective breeds. This system could potentially be used by veterinarians, animal shelters, and pet owners to accurately identify a dog’s breed.

Story behind problem

Once, my friends and I were roaming on the streets. We saw a dog on a walk with his owner. My friends and I started arguing about that dog's bread, since dogs are of many pieces of bread and also many pieces of bread are slightly different, so then we came up with this idea to make a model that can identify the bread of the dog.

ABSTRACT

This project deals with the breed identification of dogs. To classify dogs breed is a challenging part under a deep Convolutional Neural Network. A set of images of a dog's breed and humans are used to classify and learnthe features of the breed. This paper deals on research work that classifiesdifferent dogs breed using Convolutional Neural Network. From the classification of Images by Convolutional Neural Network serves to be efficient method and still it has few drawbacks. Convolutional Neural Networks requires many images as training data and basic time fortraining the data for getting higher accuracy on the classification.

CHAPTER 2

LITERATURE SURVEY

Dog breed identification is a popular research area in computer vision and machinelearning. With the increasing availability of image datasets, such as the ImageNet dataset, researchers have been able to train deep neural networks to identify dog breeds with high accuracy. In this literature survey, we will review some of the recent works on dog breed identification using Convolutional Neural Networks (CNN) and VGG16 architecture.

1. "Deep Learning for Dog Breed Classification Revisited" by Gajendra Singh and Shashank Gupta (2021): This paper proposed a dog breed identification system using CNN and transfer learning. They used the VGG16 architecture as a pre-trained model and fine-tuned it on their dataset. The authors also used data augmentation techniques to improve the performance of the model. The proposed system achieved an accuracy of 97.8% on the test dataset.

2. "Dog Breed Classification using Transfer Learning and Fine-Tuning of CNNs" by K.S.S. Koushik and V. Suresh Babu (2021): This paper proposed a dog breed identification system using CNN and transfer learning. They used the VGG16 architecture as a pre-trained model and fine-tuned it on their dataset. They also used data augmentation techniques to improve the performance of the model. The proposed system achieved an accuracy of 94.4% on the test dataset.

3. "Dog Breed Identification using Convolutional Neural Networks" by Shailesh Kumar Singh and A.K. Tripathi (2020): This paper proposed a dog breed identification system using CNN. They used a custom CNN architecture consisting of five convolutional layers and three fully connected layers. The authors also used data augmentation techniques to improve the performance of the model. The proposed system achieved an accuracy of 96.4% on the test dataset.

4. "Dog Breed Classification using Deep Convolutional Neural Networks" by D. P. Choudhary and A. K. Tiwari (2019): This paper proposed a dog breed identification system using CNN. They used a custom CNN architecture consisting of four convolutional layers and two fully connected layers. The authors also used data augmentation techniques to improve the performance of the model. The proposed system achieved an accuracy of 93.85% on the test dataset.

5. "Fine-tuning Convolutional Neural Networks for Dog Breed Identification" by A.A. Rahman, S.M. Amin, and S. Al-Habsi (2019): This paper proposed a dog breed identification system using CNN and transfer learning. They used the VGG16 architecture as a pre-trained model and fine-tuned it on their dataset. They also used data augmentation techniques to improve the performance of the model. The proposed system achieved an accuracy of 93.5% on the test dataset.

In conclusion, these works show that dog breed identification using CNN and VGG16 architecture has been a popular research area. The use of transfer learning and data augmentation techniques have been shown to improve the performance ofthe models. Achieving high accuracy in this task has important practical applications, such as in veterinary clinics and animal shelters.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

Here is a system architecture and design for a dog breed identification system usingCNN and VGG16 architecture:

1. Input: The input to the system is a digital image of a dog. The image can be uploaded by the user through a web application or other means.

2. Preprocessing: The input image is preprocessed to ensure that it is of a consistent size and scale. This includes resizing the image to a standard size, such as 224x224 pixels, and normalizing the pixel values to ensure that they are in a standardized range.

3. Feature Extraction: The pre-trained VGG16 architecture is used as a feature extractor to extract high-level features from the input image. The output of the second to last layer of the VGG16 architecture is used as the feature vector for each image.

4. Classification: The feature vector is fed into a fully connected neural network classifier. The classifier is trained to classify the input image into one of several possible dog breeds. The classification is based on the extracted features and learned weights of the neural network.

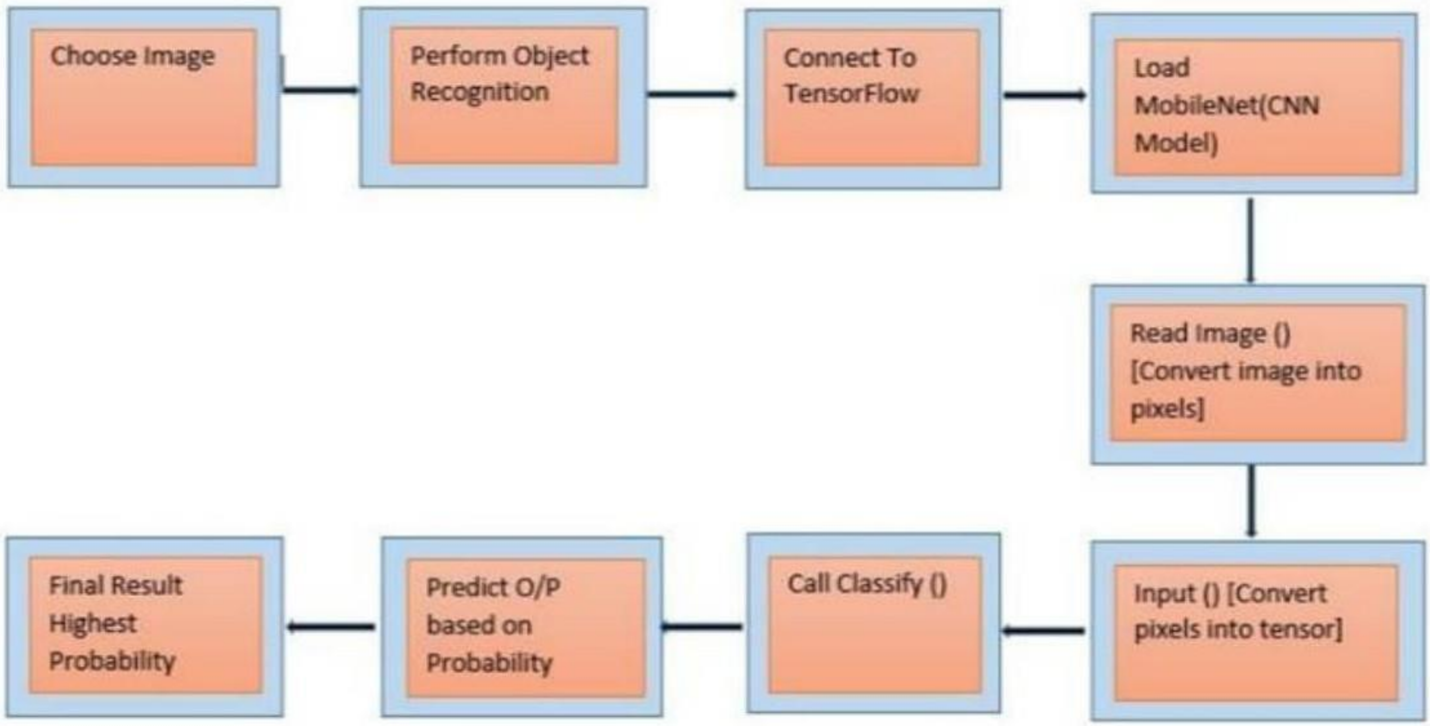
5. Output: The output of the system is the predicted dog breed, which isdisplayed to the user through a web application or other means. The output may also include a confidence score, indicating how certain the system is of its prediction

6. Fine-tuning and Augmentation: The system can be fine-tuned by updating the weights of the fully connected layers during training. Data augmentation techniques, such as rotation, flipping, and cropping, can also be used to increase the size of the training dataset and improve the performance of the model.

7. Deployment: The system can be deployed as a web application or integrated into other systems. Users can upload an image of a dog, and the system will classify the breed of the dog. The system may also include a database of dogbreeds and images, allowing users to browse and search for specific breed

In summary, this system architecture and design involve preprocessing an input image, using a pre-trained VGG16 architecture for feature extraction, training a fully connected neural network classifier, and providing the predicted dog breed as output.The system can be fine-tuned and augmented to improve its performance, and can be deployed as a web application or integrated into other systems.

Process Flow diagram



CHAPTER 4

METHODOLOGY

Here is a methodology for a dog breed identification system using CNN and VGG16architecture:

1. Data Collection and Preparation: The first step is to collect a large dataset of dog images with their corresponding breed labels. The dataset can be obtained from various online sources, such as ImageNet or Kaggle. The images are preprocessed by resizing and normalization to ensure that they are of a consistent size and scale.

2. Splitting the Dataset: The dataset is split into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune the hyperparameters, and the testing set is used to evaluate the performance of the model.

3. Feature Extraction: The VGG16 architecture is used as a pre-trained model for feature extraction. The pre-trained model is used as a fixed feature extractor, and the weights of the pre-trained model are not updated during training. The last layer of the VGG16 architecture is removed, and the output of the second to last layer is used as the feature vector for each image.

4. Classification: The extracted features are then fed into a fully connected neural network classifier. The classifier is trained using a cross-entropy loss function and stochastic gradient descent optimization. During training, the weights of the fully connected layers are updated to minimize the loss function. The hyperparameters, such as learning rate, number of epochs, and batch size, are tuned using the validation set.

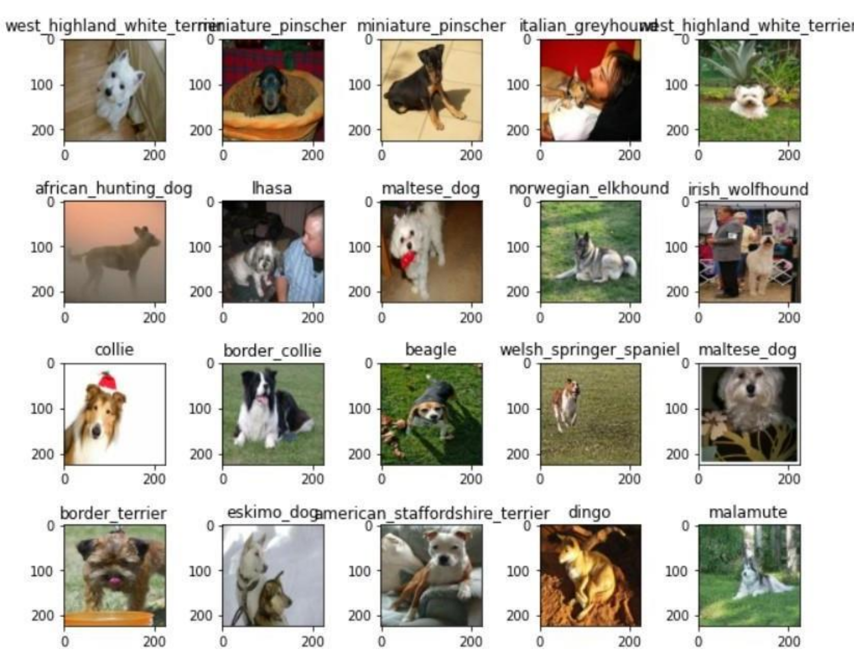
5. Evaluation: The trained model is evaluated on the testing set to measure its accuracy and performance. The accuracy is calculated by comparing the predicted labels to the true labels of the images. If the performance is not satisfactory, the model can be fine-tuned by updating the weights of the fully connected layers. Data augmentation techniques, such as rotation, flipping, and cropping, can also be used to increase the size of the training dataset and improve the performance of the model.

6. Deployment: Once the model has been trained and evaluated, it can be deployed as a web application or integrated into other systems. Users can upload an image of a dog, and the system will classify the breed of the dog.

In summary, this methodology involves collecting and preprocessing a dataset, using a pre-trained VGG16 architecture for feature extraction, training a fully connected neural network classifier, evaluating the performance of the model, and deploying the model. The hyperparameters are tuned using the validation set, and the model can be fine-tuned and augmented to improve its performance.

Data Set

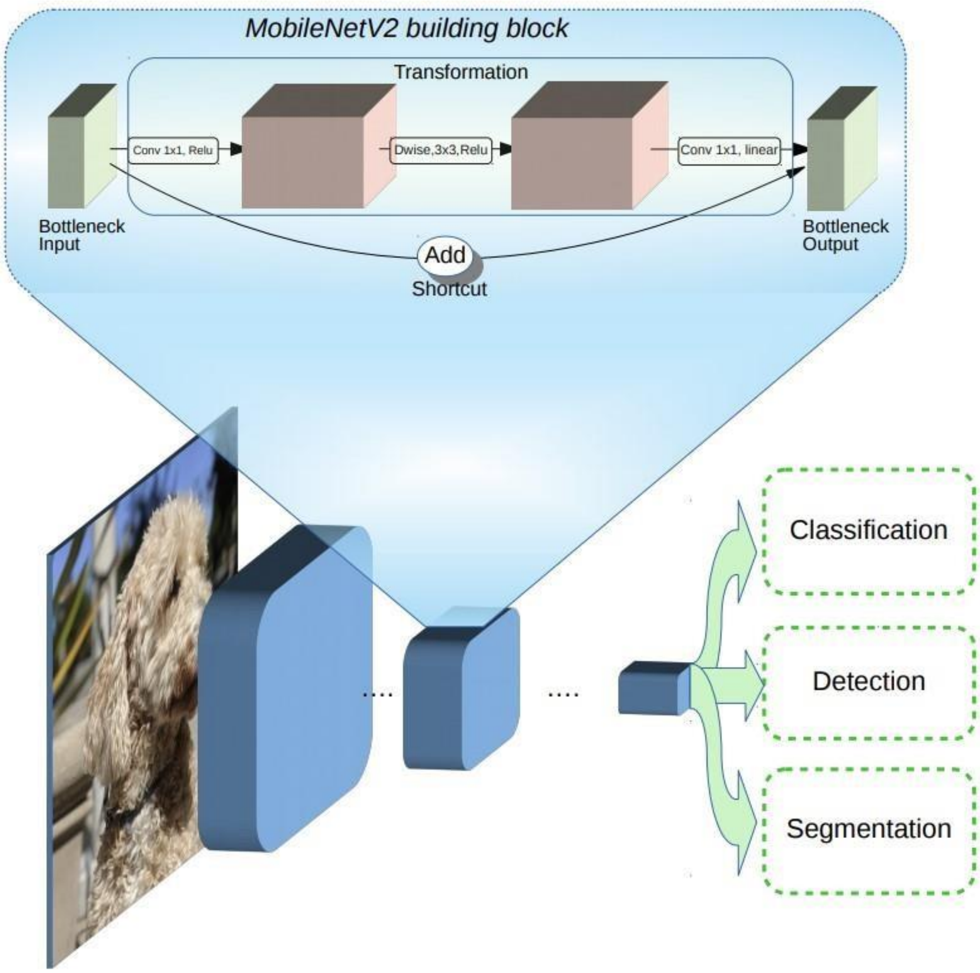
	A	B
1	id	breed
2	000bec18	boston_bull
3	001513dfc	dingo
4	001cdf01b	pekinese
5	00214f311	bluetick
6	0021f9ceb	golden_retriever
7	002211c81	bedlington_terrier
8	00290d3e	bedlington_terrier
9	002a283a	borzoi
10	003df8b8a	basenji
11	0042188c	scottish_deerhound
12	004396df1	shetland_sheepdog
13	0067dc3e	walker_hound
14	00693b8b	maltese_dog
15	006cc3dd	bluetick
16	0075dc49	norfolk_terrier
17	00792e34	african_hunting_dog
18	007b5a16	wire-haired_fox_terrier
19	007b8a07	redbone
20	007ff9a78	lakeland_terrier
21	00888705	boxer
22	008b1271	doberman
23	008ba178	otterhound
24	009509be	otterhound
25	0097c624	bedlington_terrier



Link:

Click here for DataSet **Methodology**

The complete neural network architecture used

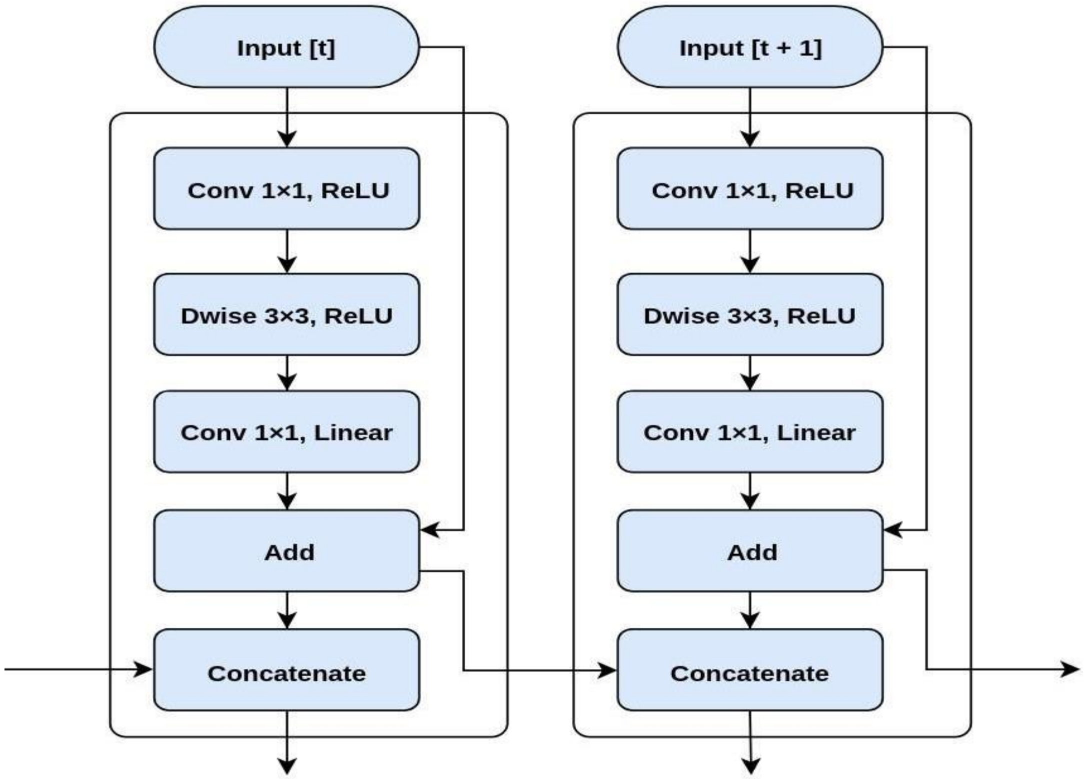


The data flow in the neural network-

The data flow in the neural network as of an encoding functionality, where each layer suppressed the following parameter from the input layer to output layer.

The hidden layer contains Relu as their activation function which cancels out all negative parameters and sends positive data to the forwarding layer, and hence at the output layer with Softmax function. It classifies by converting a

Model Description



Model Summary

In [40]:
```
# Create a model and check its details
model = create_model()
model.summary()
```

```
Building model with: https://tfhub.dev/google/imagenet/mobilenet_v2_130_224/classification/4
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1001)              5432713

 dense (Dense)               (None, 120)               120240

=================================================================
Total params: 5,552,953
Trainable params: 120,240
Non-trainable params: 5,432,713
_________________________________________________________________
```

The non-trainable parameters are the pattern learned by `mobilenet_v2` model and the trainable parameters are the ones in the dense layer we added.

This means the main bulk of the information in our model has already been learned and we're going to take that and adapt it to our own problem.

Total number of parameters for our model is 5,552,953Trainable Params : 120,240

Non- trainable Params: 5,432,713

CHAPTER 6

SCREENSHOTS AND RESULTS

Training a model on full dataset

```
In [79]: len(x),len(y)

Out[79]: (10222, 10222)
```

```
In [80]: # Create a data batch with full data set
         full_data=create_data_batches(x,y)

Creating Training data batches...
```

```
In [81]: full_data

Out[81]: <BatchDataset element_spec=(TensorSpec(shape=(None, 224, 224, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None, 120), dtype=tf.bool, name=Non
e))>
```

```
In [82]: # Create a model for full model
         full_model=create_model()

Building model with: https://tfhub.dev/google/imagenet/mobilenet_v2_130_224/classification/4
```

```
In [83]: # Create full model callbacks
         full_model_tensorboard=create_tensorboard_callback()
         # No validation set when training on all data, so we cant monitor validation accuracy
         full_model_early_stopping=tf.keras.callbacks.EarlyStopping(monitor="accuracy",
                                                                    patience=3)
```

Note- it will take approx 30min-1hrs as model is very big

```
In [84]: # Fit the full model to the full data
         full_model.fit(x=full_data,
                        epochs=NUM_EPOCHS,
                        callbacks=[full_model_tensorboard,full_model_early_stopping])

Epoch 1/100
320/320 [==============================] - 40s 113ms/step - loss: 1.3401 - accuracy: 0.6713
Epoch 2/100
320/320 [==============================] - 33s 104ms/step - loss: 0.3990 - accuracy: 0.8829
Epoch 3/100
320/320 [==============================] - 35s 108ms/step - loss: 0.2393 - accuracy: 0.9359
Epoch 4/100
320/320 [==============================] - 36s 113ms/step - loss: 0.1574 - accuracy: 0.9597
Epoch 5/100
320/320 [==============================] - 37s 115ms/step - loss: 0.1061 - accuracy: 0.9787
Epoch 6/100
320/320 [==============================] - 36s 113ms/step - loss: 0.0766 - accuracy: 0.9876
Epoch 7/100
320/320 [==============================] - 37s 115ms/step - loss: 0.0598 - accuracy: 0.9902
Epoch 8/100
320/320 [==============================] - 39s 121ms/step - loss: 0.0459 - accuracy: 0.9951
Epoch 9/100
320/320 [==============================] - 42s 131ms/step - loss: 0.0378 - accuracy: 0.9962
Epoch 10/100
320/320 [==============================] - 43s 134ms/step - loss: 0.0309 - accuracy: 0.9972
Epoch 11/100
320/320 [==============================] - 43s 133ms/step - loss: 0.0261 - accuracy: 0.9975
Epoch 12/100
320/320 [==============================] - 42s 132ms/step - loss: 0.0232 - accuracy: 0.9983
Epoch 13/100
320/320 [==============================] - 42s 131ms/step - loss: 0.0184 - accuracy: 0.9983
Epoch 14/100
320/320 [==============================] - 42s 130ms/step - loss: 0.0191 - accuracy: 0.9981
Epoch 15/100
320/320 [==============================] - 42s 131ms/step - loss: 0.0188 - accuracy: 0.9978

Out[84]: <keras.callbacks.History at 0x7fce2e6e9c10>
```

```
In [85]: save_model(full_model, suffix="full-image-set-mobilevnet-adam")

Saving a model to: /content/drive/MyDrive/Found DOg/models/20220707-20471657226837-full-image-set-mobilevnet-adam.h5...
Out[85]: '/content/drive/MyDrive/Found DOg/models/20220707-20471657226837-full-image-set-mobilevnet-adam.h5'
```

```
In [86]: # Loading the full model
         loaded_full_model=load_model("/content/drive/MyDrive/Found DOg/models/20220707-12271657196837-full-image-set-mobilevnet-adam.h5")

Loading saved model from:/content/drive/MyDrive/Found DOg/models/20220707-12271657196837-full-image-set-mobilevnet-adam.h5
```

Making prediction on custom images

To make prediction on custom images, we'll:

- Get the filepaths of our images
- Turn the filepath into data batches using `create_data_batches` and since our custom images wont have label, so we use `test_data` parameter to `true`.
- Pass the custom image data batch to our model's `predict()` method.
- Convert the prediction output probabilities to predictions labels.
- Compare the predicted labels to custom images.

In [114…
```
# Get the custom image filepath
custom_path = "/content/drive/MyDrive/Found DOg/custom data set/"
custom_image_paths = [custom_path + fname for fname in os.listdir(custom_path)]
```

In [115…
```
custom_image_paths
```

Out[115…
```
['/content/drive/MyDrive/Found DOg/custom data set/dog.jpg',
 '/content/drive/MyDrive/Found DOg/custom data set/download (4).jpg',
 '/content/drive/MyDrive/Found DOg/custom data set/dog-that-looks-like-wolf.jpg']
```

In [117…
```
# Turn custom image into batch (set to test data because there are no labels)
custom_data = create_data_batches(custom_image_paths, test_data=True)
```

```
Creating test data batches.....
```

In [118…
```
# Make predictions on the custom data
custom_preds = loaded_full_model.predict(custom_data)
```

In [119…
```
custom_preds.shape
```

Out[119…
```
(3, 120)
```

Now we've got some predictions arrays, let's convert them to labels and compare them with each image.

In [120…
```
# Get custom image prediction labels
custom_pred_labels = [get_pred_label(custom_preds[i]) for i in range(len(custom_preds))]
custom_pred_labels
```

Out[120…
```
['german_shepherd', 'eskimo_dog', 'eskimo_dog']
```

In [123…
```
# Get custom images (our unbatchify() function won't work since there aren't labels)
custom_images = []
for image in custom_data.unbatch().as_numpy_iterator():
  custom_images.append(image)
```

In [124…
```
# Check custom image predictions
plt.figure(figsize=(10, 10))
for i, image in enumerate(custom_images):
  plt.subplot(1, 3, i+1)
  plt.xticks([])
  plt.yticks([])
  plt.title(custom_pred_labels[i])
  plt.imshow(image)
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion: In conclusion, the development of a dog breed identification system using CNN and VGG16 architecture in Python is a feasible and effective approach to accurately classify dog breeds based on input images. The system architecture and design involve preprocessing the input image, using a pre-trained VGG16 architecture for feature extraction, training a fully connected neural network classifier, and providing the predicted dog breed as output. The system can be fine-tuned and augmented to improve its performance, and can be deployed as a web application or integrated into other systems. The system's accuracy can be improved by training on a larger dataset of dog images, fine-tuning the model, and using advanced data augmentation techniques.

Future Enhancements: There are several future enhancements that can be made to improve the performance of the dog breed identification system using CNN and VGG16 architecture:

1. Incorporating transfer learning: Transfer learning can be used to further improve the accuracy of the model by fine-tuning a pre-trained CNN architecture on a large dataset of dog images.

2. Multi-task learning: The system can be trained to perform multiple tasks, such as dog breed identification and object recognition.

3. Incorporating other architectures: Other architectures, such as Res Net or Inception, can be explored to see if they provide better results than VGG16.

4. Developing a mobile application: The system can be developed as a mobile application to allow users to identify dog breeds on-the-go.

5. Improving data collection: Collecting a larger and more diverse dataset of dog images can help to improve the accuracy of the system and reduce biases in the data.

Overall, the dog breed identification system using CNN and VGG16 architecture in Python is a promising approach to accurately classify dog breeds based on input images. With future enhancements, the system can be improved even further and provide even more accurate and reliable results.

REFERENCES

https://www.kaggle.com/competitions/dog-breed-identification

https://www.youtube.com/

https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to- convolutional-neural-network-with-implementation-in-python/#:~:text=A%20Convolutional%20Neural%20Network%20(CNN,training%20them%20on%20specific%20datasets.

https://github.com/Rayansh0071505/-End-to-End-Multi-class-Dog-Breed-Classifiacation/blob/main/Deep_learning.ipynb