

Final Project Report Template

1. Introduction

1.1. Project overviews

- The project basically classifies the dog breeds using transfer learning. In this project CNN Architectures like VGG-16, Resnet-50, Inception and Xception.

1.2. Objectives

- The objective of the project is to classify the different breeds of dogs so as to solve the real-world problems.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement

- The problem statement for the following project can be any of the below two:
 - To create an online platform to categorize the dog breed available for adoption based on the uploaded image.
 - A veterinarian needs assistance in identifying the breed of the dog brought in for health checkup.

2.2. Project Proposal (Proposed Solution)

Project Overview	
Objective	The objective of the project is to classify and identify the dog breed from images using transfer learning.
Scope	The project has a wider scope. The model can identify the provided 8 breeds of dog. To identify more breeds, we will need larger dataset.
Problem Statement	
Description	The problem statement that we worked on is Dog Breed Identification using the Transfer learning.
Impact	Solving the problem can make the users identify the dog breed accurately without any discomfort.
Proposed Solution	
Approach	The images are taken as input and the breed of the dog is identified. Different CNN architectures such as VGG-16, Resnet50, Inception and Xception were used to identify the breed. Among which Xception gave the best accuracy. So deployed the application with that model.
Key Features	The accuracy of the model is around 99.9% which makes the solution accurate and precise.

2.3. Initial Project Planning

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Project Initiation and Planning	USN-1	Project is initiated and the planning is done	2	High	Akshwin, Harshith Sallangi, Prashanth Kumar, Preneetha Nissy Dasari	9 July 2024	9 July 2024
Sprint-2	Data Collection and Preprocessing Phase	USN-2	Data for the project is collected from Kaggle. It contains images of eight breeds of dog.	1	High	Akshwin, Harshith Sallangi	9 July 2024	10 July 2024
Sprint-3	Model development	USN-3	The transfer learning is used to build the models. VGG-16,	2	High	Akshwin, Prashanth Kumar	10 July 2024	11 July 2024
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
			Resnet, Inception, Xception are used for it.					
Sprint-4	Hyperparameter tuning	USN-4	Hyperparameter tuning is done by changing the optimizer and by changing the epochs	2	Medium	Akshwin, Preneetha Nissy Dasari	11 July 2024	11 July 2024
Sprint-5	Application Development	USN-5	The application is developed in Flask and the model is deployed in it to make predictions.	1	High	Akshwin, Harshith Sallangi	12 July 2024	12 July 2024

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

Section	Description
Project Overview	The project identifies the breed of the dog when the image of the dog is uploaded as an input.
Data Collection Plan	The dataset has been collected from Kaggle.
Raw Data Sources Identified	The dataset is from Kaggle. It contains 8 different classes of breed.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset 1	It contains 8 classes of dog breeds	https://www.kaggle.com/datasets/mohamedchahed/dog-breeds	Image	86 MB	Public

3.2. Data Quality Report

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle	There are different number of images for different dog breeds	Low	Random function is used to separate the testing and training data which makes sure it is evenly distributed.

3.3. Data Preprocessing

Section	Description
Data Overview	The dataset is from Kaggle. It contains 541 images with 8 classes. The eight classes of breed of dog are beagle, bulldog, dalmatian, german-shepherd, husky, labrador-retriever, poodle, rottweiler
Resizing	The image is resized into a target size of 224 x 224 x 3.
Normalization	Normalized pixel value between 0 to 1.
Data Augmentation	Applied Data augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.

Data Preprocessing Code Screenshots

Loading Data	<pre># download dataset !kaggle datasets download -d 'mohamedchahed/dog-breeds' # unzip dataset !unzip dog-breeds.zip</pre>
--------------	---

Resizing	<pre># Define the image dimensions and batch size img_height = 224 img_width = 224</pre>
----------	--

Normalization	<pre>train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True) test_datagen = ImageDataGenerator(rescale=1./255)</pre>
---------------	--

Data Augmentation	<pre>train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True) test_datagen = ImageDataGenerator(rescale=1./255)</pre>
-------------------	--

4. Model Development Phase

4.1. Model Selection Report

Model	Description
Model 1	This model is build using the VGG-16 architecture by applying transfer learning. The top layer is replaced with the dense layer with 8 neurons and sigmoid activation function. The model got an accuracy of 100 for 10 epochs.
Model 2	This model is build using the ResNet-50 architecture by applying transfer learning. The top layer is replaced with the dense layer with 8 neurons and sigmoid activation function. The model got an accuracy of 42 for 10 epochs.

Model 3	This model is build using the Inception architecture by applying transfer learning. The top layer is replaced with the dense layer with 8 neurons and sigmoid activation function. The model got an accuracy of 28.5 for 10 epochs.
Model 4	This model is build using the Xception architecture by applying transfer learning. The top layer is replaced with the dense layer with 8 neurons and sigmoid activation function. The model got an accuracy of 100 for 10 epochs.

4.2. Initial Model Training Code, Model Validation and Evaluation Report

Initial Model Training

```
vgg16.fit(train_generator,validation_data = test_generator,epochs=10 )
```

```
resnet.fit(train_generator,validation_data = test_generator,epochs=10 )
```

```
inception.fit(train_generator,validation_data = test_generator,epochs=10 )
```

```
xception.fit(train_generator,validation_data = test_generator,epochs=10 )
```

Model Validation and Evaluation Report

Model

Summary

Model 1

Model: Sumo1

Layer Type	Output Size	Param #
Input (Embedding)	[256, 64, 128, 32]	0
Block 1 (Conv2D)	[64, 128, 128, 64]	1792
Block 2 (Conv2D)	[64, 128, 128, 64]	1792
Block 3 (ReLU)	[64, 128, 128, 64]	0
Block 4 (Conv2D)	[128, 128, 128, 128]	1792
Block 5 (Conv2D)	[128, 128, 128, 128]	1792
Block 6 (ReLU)	[128, 128, 128, 128]	0
Block 7 (Conv2D)	[64, 64, 64, 64]	1792
Block 8 (Conv2D)	[64, 64, 64, 64]	1792
Block 9 (ReLU)	[64, 64, 64, 64]	0
Block 10 (Conv2D)	[64, 64, 64, 64]	1792
Block 11 (Conv2D)	[64, 64, 64, 64]	1792
Block 12 (ReLU)	[64, 64, 64, 64]	0
Block 13 (Conv2D)	[64, 64, 64, 64]	1792
Block 14 (Conv2D)	[64, 64, 64, 64]	1792
Block 15 (ReLU)	[64, 64, 64, 64]	0
Block 16 (Conv2D)	[64, 64, 64, 64]	1792
Block 17 (Conv2D)	[64, 64, 64, 64]	1792
Block 18 (ReLU)	[64, 64, 64, 64]	0
Block 19 (Conv2D)	[64, 64, 64, 64]	1792
Block 20 (Conv2D)	[64, 64, 64, 64]	1792
Block 21 (ReLU)	[64, 64, 64, 64]	0
Block 22 (Conv2D)	[64, 64, 64, 64]	1792
Block 23 (Conv2D)	[64, 64, 64, 64]	1792
Block 24 (ReLU)	[64, 64, 64, 64]	0
Block 25 (Conv2D)	[64, 64, 64, 64]	1792
Block 26 (Conv2D)	[64, 64, 64, 64]	1792
Block 27 (ReLU)	[64, 64, 64, 64]	0
Block 28 (Conv2D)	[64, 64, 64, 64]	1792
Block 29 (Conv2D)	[64, 64, 64, 64]	1792
Block 30 (ReLU)	[64, 64, 64, 64]	0
Block 31 (Conv2D)	[64, 64, 64, 64]	1792
Block 32 (Conv2D)	[64, 64, 64, 64]	1792
Block 33 (ReLU)	[64, 64, 64, 64]	0
Block 34 (Conv2D)	[64, 64, 64, 64]	1792
Block 35 (Conv2D)	[64, 64, 64, 64]	1792
Block 36 (ReLU)	[64, 64, 64, 64]	0
Block 37 (Conv2D)	[64, 64, 64, 64]	1792
Block 38 (Conv2D)	[64, 64, 64, 64]	1792
Block 39 (ReLU)	[64, 64, 64, 64]	0
Block 40 (Conv2D)	[64, 64, 64, 64]	1792
Block 41 (Conv2D)	[64, 64, 64, 64]	1792
Block 42 (ReLU)	[64, 64, 64, 64]	0
Block 43 (Conv2D)	[64, 64, 64, 64]	1792
Block 44 (Conv2D)	[64, 64, 64, 64]	1792
Block 45 (ReLU)	[64, 64, 64, 64]	0
Block 46 (Conv2D)	[64, 64, 64, 64]	1792
Block 47 (Conv2D)	[64, 64, 64, 64]	1792
Block 48 (ReLU)	[64, 64, 64, 64]	0
Block 49 (Conv2D)	[64, 64, 64, 64]	1792
Block 50 (Conv2D)	[64, 64, 64, 64]	1792
Block 51 (ReLU)	[64, 64, 64, 64]	0
Block 52 (Conv2D)	[64, 64, 64, 64]	1792
Block 53 (Conv2D)	[64, 64, 64, 64]	1792
Block 54 (ReLU)	[64, 64, 64, 64]	0
Block 55 (Conv2D)	[64, 64, 64, 64]	1792
Block 56 (Conv2D)	[64, 64, 64, 64]	1792
Block 57 (ReLU)	[64, 64, 64, 64]	0
Block 58 (Conv2D)	[64, 64, 64, 64]	1792
Block 59 (Conv2D)	[64, 64, 64, 64]	1792
Block 60 (ReLU)	[64, 64, 64, 64]	0
Block 61 (Conv2D)	[64, 64, 64, 64]	1792
Block 62 (Conv2D)	[64, 64, 64, 64]	1792
Block 63 (ReLU)	[64, 64, 64, 64]	0
Block 64 (Conv2D)	[64, 64, 64, 64]	1792
Block 65 (Conv2D)	[64, 64, 64, 64]	1792
Block 66 (ReLU)	[64, 64, 64, 64]	0
Block 67 (Conv2D)	[64, 64, 64, 64]	1792
Block 68 (Conv2D)	[64, 64, 64, 64]	1792
Block 69 (ReLU)	[64, 64, 64, 64]	0
Block 70 (Conv2D)	[64, 64, 64, 64]	1792
Block 71 (Conv2D)	[64, 64, 64, 64]	1792
Block 72 (ReLU)	[64, 64, 64, 64]	0
Block 73 (Conv2D)	[64, 64, 64, 64]	1792
Block 74 (Conv2D)	[64, 64, 64, 64]	1792
Block 75 (ReLU)	[64, 64, 64, 64]	0
Block 76 (Conv2D)	[64, 64, 64, 64]	1792
Block 77 (Conv2D)	[64, 64, 64, 64]	1792
Block 78 (ReLU)	[64, 64, 64, 64]	0
Block 79 (Conv2D)	[64, 64, 64, 64]	1792
Block 80 (Conv2D)	[64, 64, 64, 64]	1792
Block 81 (ReLU)	[64, 64, 64, 64]	0
Block 82 (Conv2D)	[64, 64, 64, 64]	1792
Block 83 (Conv2D)	[64, 64, 64, 64]	1792
Block 84 (ReLU)	[64, 64, 64, 64]	0
Block 85 (Conv2D)	[64, 64, 64, 64]	1792
Block 86 (Conv2D)	[64, 64, 64, 64]	1792
Block 87 (ReLU)	[64, 64, 64, 64]	0
Block 88 (Conv2D)	[64, 64, 64, 64]	1792
Block 89 (Conv2D)	[64, 64, 64, 64]	1792
Block 90 (ReLU)	[64, 64, 64, 64]	0
Block 91 (Conv2D)	[64, 64, 64, 64]	1792
Block 92 (Conv2D)	[64, 64, 64, 64]	1792
Block 93 (ReLU)	[64, 64, 64, 64]	0
Block 94 (Conv2D)	[64, 64, 64, 64]	1792
Block 95 (Conv2D)	[64, 64, 64, 64]	1792
Block 96 (ReLU)	[64, 64, 64, 64]	0
Block 97 (Conv2D)	[64, 64, 64, 64]	1792
Block 98 (Conv2D)	[64, 64, 64, 64]	1792
Block 99 (ReLU)	[64, 64, 64, 64]	0
Block 100 (Conv2D)	[64, 64, 64, 64]	1792
Block 101 (Conv2D)	[64, 64, 64, 64]	1792
Block 102 (ReLU)	[64, 64, 64, 64]	0
Block 103 (Conv2D)	[64, 64, 64, 64]	1792
Block 104 (Conv2D)	[64, 64, 64, 64]	1792
Block 105 (ReLU)	[64, 64, 64, 64]	0
Block 106 (Conv2D)	[64, 64, 64, 64]	1792
Block 107 (Conv2D)	[64, 64, 64, 64]	1792
Block 108 (ReLU)	[64, 64, 64, 64]	0
Block 109 (Conv2D)	[64, 64, 64, 64]	1792
Block 110 (Conv2D)	[64, 64, 64, 64]	1792
Block 111 (ReLU)	[64, 64, 64, 64]	0
Block 112 (Conv2D)	[64, 64, 64, 64]	1792
Block 113 (Conv2D)	[64, 64, 64, 64]	1792
Block 114 (ReLU)	[64, 64, 64, 64]	0
Block 115 (Conv2D)	[64, 64, 64, 64]	1792
Block 116 (Conv2D)	[64, 64, 64, 64]	1792
Block 117 (ReLU)	[64, 64, 64, 64]	0
Block 118 (Conv2D)	[64, 64, 64, 64]	1792
Block 119 (Conv2D)	[64, 64, 64, 64]	1792
Block 120 (ReLU)	[64, 64, 64, 64]	0
Block 121 (Conv2D)	[64, 64, 64, 64]	1792
Block 122 (Conv2D)	[64, 64, 64, 64]	1792
Block 123 (ReLU)	[64, 64, 64, 64]	0
Block 124 (Conv2D)	[64, 64, 64, 64]	1792
Block 125 (Conv2D)	[64, 64, 64, 64]	1792
Block 126 (ReLU)	[64, 64, 64, 64]	0
Block 127 (Conv2D)	[64, 64, 64, 64]	1792
Block 128 (Conv2D)	[64, 64, 64, 64]	1792
Block 129 (ReLU)	[64, 64, 64, 64]	0
Block 130 (Conv2D)	[64, 64, 64, 64]	1792
Block 131 (Conv2D)	[64, 64, 64, 64]	1792
Block 132 (ReLU)	[64, 64, 64, 64]	0
Block 133 (Conv2D)	[64, 64, 64, 64]	1792
Block 134 (Conv2D)	[64, 64, 64, 64]	1792
Block 135 (ReLU)	[64, 64, 64, 64]	0
Block 136 (Conv2D)	[64, 64, 64, 64]	1792
Block 137 (Conv2D)	[64, 64, 64, 64]	1792
Block 138 (ReLU)	[64, 64, 64, 64]	0
Block 139 (Conv2D)	[64, 64, 64, 64]	1792
Block 140 (Conv2D)	[64, 64, 64, 64]	1792
Block 141 (ReLU)	[64, 64, 64, 64]	0
Block 142 (Conv2D)	[64, 64, 64, 64]	1792
Block 143 (Conv2D)	[64, 64, 64, 64]	1792
Block 144 (ReLU)	[64, 64, 64, 64]	0
Block 145 (Conv2D)	[64, 64, 64, 64]	1792
Block 146 (Conv2D)	[64, 64, 64, 64]	1792
Block 147 (ReLU)	[64, 64, 64, 64]	0
Block 148 (Conv2D)	[64, 64, 64, 64]	1792
Block 149 (Conv2D)	[64, 64, 64, 64]	1792
Block 150 (ReLU)	[64, 64, 64, 64]	0
Block 151 (Conv2D)	[64, 64, 64, 64]	1792
Block 152 (Conv2D)	[64, 64, 64, 64]	1792
Block 153 (ReLU)	[64, 64, 64, 64]	0
Block 154 (Conv2D)	[64, 64, 64, 64]	1792
Block 155 (Conv2D)	[64, 64, 64, 64]	1792
Block 156 (ReLU)	[64, 64, 64, 64]	0
Block 157 (Conv2D)	[64, 64, 64, 64]	1792
Block 158 (Conv2D)	[64, 64, 64, 64]	1792
Block 159 (ReLU)	[64, 64, 64, 64]	0
Block 160 (Conv2D)	[64, 64, 64, 64]	1792
Block 161 (Conv2D)	[64, 64, 64, 64]	1792
Block 162 (ReLU)	[64, 64, 64, 64]	0
Block 163 (Conv2D)	[64, 64, 64, 64]	1792
Block 164 (Conv2D)	[64, 64, 64, 64]	1792
Block 165 (ReLU)	[64, 64, 64, 64]	0
Block 166 (Conv2D)	[64, 64, 64, 64]	1792
Block 167 (Conv2D)	[64, 64, 64, 64]	1792
Block 168 (ReLU)	[64, 64, 64, 64]	0
Block 169 (Conv2D)	[64, 64, 64, 64]	1792
Block 170 (Conv2D)	[64, 64, 64, 64]	1792
Block 171 (ReLU)	[64, 64, 64, 64]	0
Block 172 (Conv2D)	[64, 64, 64, 64]	1792
Block 173 (Conv2D)	[64, 64, 64, 64]	1792
Block 174 (ReLU)	[64, 64, 64, 64]	0
Block 175 (Conv2D)	[64, 64, 64, 64]	1792
Block 176 (Conv2D)	[64, 64, 64, 64]	1792
Block 177 (ReLU)	[64, 64, 64, 64]	0
Block 178 (Conv2D)	[64, 64, 64, 64]	1792
Block 179 (Conv2D)	[64, 64, 64, 64]	1792
Block 180 (ReLU)	[64, 64, 64, 64]	0
Block 181 (Conv2D)	[64, 64, 64, 64]	1792
Block 182 (Conv2D)	[64, 64, 64, 64]	1792
Block 183 (ReLU)	[64, 64, 64, 64]	0
Block 184 (Conv2D)	[64, 64, 64, 64]	1792
Block 185 (Conv2D)	[64, 64, 64, 64]	1792
Block 186 (ReLU)	[64, 64, 64, 64]	0
Block 187 (Conv2D)	[64, 64, 64, 64]	1792
Block 188 (Conv2D)	[64, 64, 64, 64]	1792
Block 189 (ReLU)	[64, 64, 64, 64]	0
Block 190 (Conv2D)	[64, 64, 64, 64]	1792
Block 191 (Conv2D)	[64, 64, 64, 64]	1792
Block 192 (ReLU)	[64, 64, 64, 64]	0
Block 193 (Conv2D)	[64, 64, 64, 64]	1792
Block 194 (Conv2D)	[64, 64, 64, 64]	1792
Block 195 (ReLU)	[64, 64, 64, 64]	0
Block 196 (Conv2D)	[64, 64, 64, 64]	1792
Block 197 (Conv2D)	[64, 64, 64, 64]	1792
Block 198 (ReLU)	[64, 64, 64, 64]	0
Block 199 (Conv2D)	[64, 64, 64, 64]	1792
Block 200 (Conv2D)	[64, 64, 64, 64]	1792
Block 201 (ReLU)	[64, 64, 64, 64]	0
Block 202 (Conv2D)	[64, 64, 64, 64]	1792
Block 203 (Conv2D)	[64, 64, 64, 64]	1792
Block 204 (ReLU)	[64, 64, 64, 64]	0
Block 205 (Conv2D)	[64, 64, 64, 64]	1792
Block 206 (Conv2D)	[64, 64, 64, 64]	1792
Block 207 (ReLU)	[64, 64, 64, 64]	0
Block 208 (Conv2D)	[64, 64, 64, 64]	1792
Block 209 (Conv2D)	[64, 64, 64, 64]	1792
Block 210 (ReLU)	[64, 64, 64, 64]	0
Block 211 (Conv2D)	[64, 64, 64, 64]	1792
Block 212 (Conv2D)	[64, 64, 64, 64]	1792
Block 213 (ReLU)	[64, 64, 64, 64]	0
Block 214 (Conv2D)	[64, 64, 64, 64]	1792
Block 215 (Conv2D)	[64, 64, 64, 64]	1792
Block 216 (ReLU)	[64, 64, 64, 64]	0
Block 217 (Conv2D)	[64, 64, 64, 64]	1792
Block 218 (Conv2D)	[64, 64, 64, 64]	1792
Block 219 (ReLU)	[64, 64, 64, 64]	0
Block 220 (Conv2D)	[64, 64, 64, 64]	1792
Block 221 (Conv2D)	[64, 64, 64, 64]	1792
Block 222 (ReLU)	[64, 64, 64, 64]	0
Block 223 (Conv2D)	[64, 64, 64, 64]	1792
Block 224 (Conv2D)	[64, 64, 64, 64]	1792
Block 225 (ReLU)	[64, 64, 64, 64]	0
Block 226 (Conv2D)	[64, 64, 64, 64]	1792
Block 227 (Conv2D)	[64, 64, 64, 64]	1792
Block 228 (ReLU)	[64, 64, 64, 64]	0
Block 229 (Conv2D)	[64, 64, 64, 64]	1792
Block 230 (Conv2D)	[64, 64, 64, 64]	1792
Block 231 (ReLU)	[64, 64, 64, 64]	0
Block 232 (Conv2D)	[64, 64, 64, 64]	1792
Block 233 (Conv2D)	[64, 64, 64, 64]	1792
Block 234 (ReLU)	[64, 64, 64, 64]	0
Block 235 (Conv2D)	[64, 64, 64, 64]	1792
Block 236 (Conv2D)	[64, 64, 64, 64]	1792
Block 237 (ReLU)	[64, 64, 64, 64]	0
Block 238 (Conv2D)	[64, 64, 64, 64]	1792
Block 239 (Conv2D)	[64, 64, 64, 64]	1792
Block 240 (ReLU)	[64, 64, 64, 64]	0
Block 241 (Conv2D)	[64, 64, 64, 64]	1792
Block 242 (Conv2D)	[64, 64, 64, 64]	1792
Block 243 (ReLU)	[64, 64, 64, 64]	0
Block 244 (Conv2D)	[64, 64, 64, 64]	1792
Block 245 (Conv2D)	[64, 64, 64, 64]	1792
Block 246 (ReLU)	[64, 64, 64, 64]	0
Block 247 (Conv2D)	[64, 64, 64, 64]	1792
Block 248 (Conv2D)	[64, 64, 64, 64]	1792
Block 249 (ReLU)	[64, 64, 64, 64]	0
Block 250 (Conv2D)	[64, 64, 64, 64]	1792
Block 251 (Conv2D)	[64, 64, 64, 64]	1792
Block 252 (ReLU)	[64, 64, 64, 64]	0
Block 253 (Conv2D)	[64, 64, 64, 64]	1792
Block 254 (Conv2D)	[64, 64, 64, 64]	1792
Block 255 (ReLU)	[64, 64, 64, 64]	0
Block 256 (Conv2D)	[64, 64, 64, 64]	1792
Block 257 (Conv2D)	[64, 64, 64, 64]	1792
Block 258 (ReLU)	[64, 64, 64, 64]	0
Block 259 (Conv2D)	[64, 64, 64, 64]	1792
Block 260 (Conv2D)	[64, 64, 64, 64]	1792
Block 261 (ReLU)	[64, 64, 64, 64]	0
Block 262 (Conv2D)	[64, 64, 64, 64]	1792
Block 263 (Conv2D)	[64, 64, 64, 64]	1792
Block 264 (ReLU)	[64, 64, 64, 64]	0
Block 265 (Conv2D)	[64, 64, 64, 64]	1792
Block 266 (Conv2D)	[64, 64, 64, 64]	1792
Block 267 (ReLU)	[64, 64, 64, 64]	0
Block 268 (Conv2D)	[64, 64, 64, 64]	1792
Block 269 (Conv2D)	[64, 64, 64, 64]	1792
Block 270 (ReLU)	[64, 64, 64, 64]	0
Block 271 (Conv2D)	[64, 64, 64, 64]	1792
Block 272 (Conv2D)	[64, 64, 64, 64]	1792
Block 273 (ReLU)	[64, 64, 64, 64]	0
Block 274 (Conv2D)	[64, 64, 64, 64]	1792
Block 275 (Conv2D)	[64, 64, 64, 64]	1792
Block 276 (ReLU)	[64, 64, 64, 64]	0
Block 277 (Conv2D)	[64, 64, 64, 64]	1792
Block 278 (Conv2D)	[64, 64, 64, 64]	1792
Block 279 (ReLU)	[64, 64, 64, 64]	

5. Model Optimization and Tuning Phase

5.1. Tuning Documentation

Model	Tuned Hyperparameters
Model 1(VGG-16)	Used Adam optimizer, which gave better accuracy than SGD and ran for 10 epochs.
Model 2(ResNet50)	Used Adam optimizer, which gave better accuracy than SGD and ran for 10 epochs.
Model 3(Inception)	Used Adam optimizer, which gave better accuracy than SGD and ran for 10 epochs.
Model 4(Xception)	Used Adam optimizer, which gave better accuracy than SGD and ran for 10 epochs.

5.2. Final Model Selection Justification

Final Model	Reasoning
Model 4 (Xception)	This model gave batter accuracy than other models.

6. Results

6.1. Output Screenshots

DOG BREED IDENTIFICATION USING TRANSFER LEARNING

DOG BREED IDENTIFICATION

Please upload an animal image

Choose...

Result: the predicted breed is : german-shepherd

7. Advantages & Disadvantages

Advantages:

- Transfer learning leverages pre-trained models on large datasets (like ImageNet), allowing for quicker convergence and significantly reducing the time needed for training.
- Pre-trained models have learned rich feature representations, which can enhance the accuracy of the classification task, especially when dealing with limited data.
- Transfer learning can achieve good performance even with smaller datasets, which is beneficial if you don't have access to a large dataset of dog breeds.
- Using complex, deep networks (like ResNet, VGG) becomes feasible without the need to train them from scratch, making advanced architectures accessible.
- The features learned from a broad dataset can help the model generalize better to different types of dog breeds, improving robustness.

Disadvantages:

- Pre-trained models might not be specialized for the task of dog breed classification and may include features irrelevant to this specific task.
- There might be a difference between the source dataset (e.g., ImageNet) and the target dataset (dog breeds), causing a performance drop due to domain shift.
- Pre-trained models are often large and computationally expensive, which might not be suitable for deployment in resource-constrained environments.
- If the target dataset is very small, there's a risk of overfitting to the small dataset despite the use of pre-trained models.
- The quality of your results is heavily dependent on the pre-trained model you choose. If the pre-trained model is not well-suited to your specific task, performance can be suboptimal.

8. Conclusion

- The project uses transfer learning to identify the breed of the dog.
- The Xception architecture gave the best result.
- So, it is used for deploying in the Flask application

9. Future Scope

Enhanced Model Accuracy:

- Continued improvements in deep learning algorithms and architectures could lead to even higher accuracy in classifying dog breeds.

Real-Time Classification:

- Development of lightweight, efficient models that can run on mobile devices, enabling real-time classification through smartphone apps.

Integration with IoT:

- Combining dog breed classification with Internet of Things (IoT) devices, such as smart collars or home cameras, for continuous monitoring and identification.

Explainable AI:

- Incorporating explainability features to provide users with insights into how the model makes its decisions, increasing trust and usability.

10. Appendix

10.1. Source Code

```
import os
import shutil
import random
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_dir = 'train'
test_dir = 'test'
img_height = 299
img_width = 299
batch_size = 32
```



```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=20,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(img_height, img_width),
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    color_mode='rgb')
```

```
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size=(img_height, img_width),
                                                  batch_size=batch_size,
                                                  class_mode='categorical',
                                                  color_mode='rgb')
```

```
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import Dense , Flatten
from tensorflow.keras.models import Model
xception= Xception(include_top = False,input_shape=(299,299,3))
```

```
for layer in xception.layers:
    print(layer)
for layer in xception.layers:
    layer.trainable = False
```

```
x = Flatten()(xception.output)
output = Dense(8,activation = 'softmax')(x)

xception= Model(xception.input,output)

xception.compile(loss = 'categorical_crossentropy',optimizer = 'adam',metrics=['accuracy'])

xception.fit(train_generator,validation_data = test_generator,epochs=10 )
```

10.2. GitHub & Project Demo Link

- Github link
 - <https://github.com/akshwin/Dog-Breed-Identification-Using-Transfer-Learning>
- Project demo link
 - <https://www.youtube.com/watch?v=60b7-n83Vm8&t=16s>