

CHAPTER 1

INTRODUCTION

The "Cat vs. Dog Prediction Project" embarks on a fascinating journey into the realm of computer vision and machine learning, tackling one of the most iconic and relatable classification tasks: distinguishing between images of cats and dogs. This project encapsulates the essence of image classification, a fundamental problem in the field of artificial intelligence, and serves as a practical demonstration of how modern technology can be applied to everyday challenges.

The ability to discern between cats and dogs has long captivated the human imagination, transcending the boundaries of simple curiosity. It has implications ranging from personal pet identification to industrial automation, with applications in fields such as animal welfare, security, and more. This project is not just an exercise in image recognition but an exploration of the power and potential of deep learning, transfer learning, and data science.

The objectives of this project are manifold:

1.Data Collection and Preparation: We will amass a comprehensive dataset containing thousands of cat and dog images, ensuring diversity and quality. The dataset will be meticulously preprocessed, with the aim of producing a clean and balanced set for training and evaluation.

2.Model Development: Leveraging state-of-the-art convolutional neural networks (CNNs) and transfer learning techniques, we will design and train a robust image classification model. classification tasks.

3.Model Evaluation: To gauge the effectiveness of our classification system, we will conduct thorough model evaluations using relevant performance metrics. We will examine accuracy, precision, recall, and F1 score, among others, to understand how well the model performs in practice.

4.Interpretability and Explain ability: We will explore the inner workings of our model, striving to make its decision-making process more transparent and interpretable. This aspect is critical for understanding why the model makes certain predictions and ensures responsible AI deployment.

5.Ethical Considerations: Recognizing the broader implications of image classification, we will discuss potential biases and ethical challenges that arise in the context of pet image classification. We will consider privacy, consent, and fairness concerns, offering insights into addressing these issues responsibly.

In summary, the "Cat vs. Dog Prediction Project" is not merely an exercise in identifying pets from pictures but a multi-faceted exploration of the capabilities and responsibilities that come with deploying image classification models in the real world. By the end of this project, we aim to provide a valuable resource for the machine learning community, as well as insights into the challenges and opportunities presented by image classification technology in our daily lives.

CHAPTER 2

SYSTEM SPECIFICATION

Systems implementation is the process of: defining how the information system should be built (i.e., physical system design), ensuring that the information system is operational and used, ensuring that the information system meets quality standard (i.e., quality assurance). The implementation phase involves putting the project plan into action. It's here that the project manager will coordinate and direct project resources to meet the objectives of the project plan. As the project unfolds, it's the project manager's job to direct and manage each activity, every step of the way.

2.1 HARDWARE SPECIFICATION

Processor	:	Intel i3 2.93 GHZ
RAM	:	2 GB
Floppy	:	1.44 MB
Mouse	:	Optical Mouse
Monitor	:	SVGA
Key board	:	104 Keys Standard
Hard disk	:	500 GB

2.2 SOFTWARE SPECIFICATION

Operating System	:	Windows XP or Higher
Front End	:	Python(Deep Learning)
Data Set	:	Cat vs Dog

2.3 LANGUAGE DESCRIPTION

PYTHON

Python is a high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council

Python interpreters are available for many operating systems. A global community of programmers develops and maintains C Python, A non-profit organization, the Python Software Foundation, manages and directs resources for Python and C Python development.

Features of Python:

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming. Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at [official web address](#). The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python OOPs Concepts

Like other general purpose languages, python is also an object-oriented language since its beginning. Python is an object-oriented programming language. It allows us to develop applications using an Object Oriented approach. In Python, we can easily create and use classes and objects.

Object

The object is an entity that has state and behaviour. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc. Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute `__doc__`, which returns the doc string defined in the function source code.

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

Inheritance

Inheritance is the most important aspect of object-oriented programming which simulates the real world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object. By using inheritance,

we can create a class which uses all the properties object. By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class. It provides re-usability of the code.

Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many and Morphs means form, shape. By polymorphism, we understand that one task can be performed in different ways. For example a class having animal, and all animals speak. But they speak differently. Here, the "speak" behaviour is polymorphic in the sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Encapsulation

Encapsulation is also an important aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonym because data abstraction is achieved through encapsulation. Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does. Data abstraction is the reduction of a particular body of data to a simplified representation of the whole. Abstraction, in general, is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

Python 3.7

Python 3.7.0 it was released on June 27 my first attempts to run it on WSL (Windows Subsystem for Linux) Ubuntu didn't quite go as planned. There's no Debian or Ubuntu distribution of Python 3.7.0 at the moment, just the sources, so I used pyenv, which fetches the sources and build them. There's a lot of new stuff in Python 3.7.0, much of it quite simple.

Features of Python 3.7.0

- The breakpoint() Built-In.
- Data Classes.
- Customization of Module Attributes.
- Typing Enhancements.
- Timing Precision.

Deep Learning

Deep learning is a subfield of machine learning that focuses on training artificial neural networks to perform tasks that typically require human intelligence. It has gained significant attention and success in recent years due to its ability to handle complex, large-scale, and unstructured data, making it particularly well-suited for tasks like image and speech recognition, natural language processing, and many other applications. Here are key aspects of deep learning:

Artificial Neural Networks (ANNs): Deep learning is built on artificial neural networks, computational models inspired by the structure and function of the human brain. ANNs consist of interconnected nodes (neurons) organized into layers, including input, hidden, and output layers.

Deep learning networks have multiple hidden layers, which allows them to capture complex patterns in data.

Feature Learning: One of the strengths of deep learning is its capacity to automatically learn and extract relevant features from raw data. In contrast to traditional machine learning, where features are often engineered by humans, deep learning models can discover and adapt their own features.

Training and Backpropagation: Deep learning models are trained through a process known as backpropagation. During training, the network makes predictions, computes an error or loss based on the difference between its predictions and the actual target values, and then adjusts its internal parameters (weights and biases) to minimize this error.

Activation Functions: Activation functions introduce non-linearity into neural networks, allowing them to model complex relationships within data. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.

Convolutional Neural Networks (CNNs): CNNs are a type of deep learning network specifically designed for tasks involving grid-like data, such as images. They use convolutional layers to automatically extract hierarchical features from images, making them highly effective for image classification and object detection.

Recurrent Neural Networks (RNNs): RNNs are used for tasks involving sequential data, such as natural language processing and time series analysis. They have a memory element that allows them to consider context from previous time steps.

Applications: Deep learning has applications in a wide range of fields, including:

Computer Vision: Image classification, object detection, facial recognition.

Natural Language Processing: Sentiment analysis, machine translation, chatbots.

Speech Recognition: Virtual assistants, transcription services.

Healthcare: Disease diagnosis, medical image analysis, drug discovery.

Autonomous Vehicles: Self-driving cars and drones.

Recommender Systems: Personalized content and product recommendations.

Challenges: Deep learning requires substantial computational resources, large labeled datasets, and can be susceptible to overfitting. However, researchers are continually addressing these challenges to make deep learning more accessible and effective.

Frameworks and Tools: There are popular deep learning frameworks and libraries like TensorFlow, PyTorch, and Keras that simplify the development and deployment of deep learning models.

Deep learning has had a transformative impact on various industries, pushing the boundaries of what's possible in terms of automation, pattern recognition, and data analysis. Its remarkable capabilities make it a powerful tool for solving complex problems across a wide range of domains.

CHAPTER 3

SYSTEM ANALYSIS

Objective: To develop a deep learning system capable of accurately classifying images as either "car" or "dog" for various real-world applications, such as autonomous vehicles, security, and pet management.

Scope: The system will involve image data collection, preprocessing, model development, training, evaluation, and potentially real-time inference.

1. System Requirements:

Data Collection and Preprocessing:

- ✓ Acquire a diverse and balanced dataset of labeled images containing cars and dogs.
- ✓ Preprocess the data to ensure uniform size, resolution, and format.
- ✓ Annotate the dataset with accurate labels.
- ✓ Split the data into training, validation, and testing sets.

Model Development:

- ✓ Choose an appropriate deep learning architecture (e.g., Convolutional Neural Network).
- ✓ Design the neural network with input, hidden, and output layers.
- ✓ Configure model parameters, including activation functions, number of layers, and optimizer.

- ✓ Implement data augmentation techniques to enhance model generalization.
- ✓ Consider transfer learning with pre-trained models for improved performance.

Training and Evaluation:

- ✓ Train the model on the training dataset.
- ✓ Monitor training progress and fine-tune hyperparameters.
- ✓ Evaluate the model using standard performance metrics (e.g., accuracy, precision, recall, F1 score).
- ✓ Implement model explainability techniques to understand decision-making.

Deployment:

- ✓ Develop an application or API for real-time inference (if applicable).
- ✓ Ensure compatibility with target platforms (e.g., cloud servers, edge devices).
- ✓ Address privacy and ethical concerns, considering biases in model predictions.

2. Data Flow:

Data Collection: Acquire images of cars and dogs from various sources, ensuring diversity and quality.

Data Preprocessing: Clean, resize, and normalize the data to prepare it for model training.

Model Development: Create a deep learning model, selecting an appropriate architecture and configuring hyperparameters.

Training: Train the model on the labeled training dataset, optimizing for performance.

Evaluation: Assess the model's performance on the validation and test datasets.

Deployment: If required, deploy the model to an application, API, or platform for real-world use.

3. Technology Stack:

Deep Learning Frameworks: TensorFlow, PyTorch, or Keras.

Programming Languages: Python for data preprocessing and model development.

Deployment: Docker containers for scalability and cloud platforms like AWS, Azure, or Google Cloud.

4. System Constraints and Risks:

Hardware Requirements: Consider the computational resources needed for training and inference.

Data Privacy: Address data privacy concerns and ensure compliance with relevant regulations.

Model Bias: Guard against biases in predictions, particularly for real-world applications.

5. Project Timeline:

Define a project schedule with milestones for data collection, model development, training, evaluation, and deployment.

Allocate time for testing, troubleshooting, and potential revisions.

6. System Maintenance and Updates:

Develop a plan for ongoing maintenance, including regular updates to the model to adapt to changing data distributions and improve accuracy.

Monitor the system's performance and address issues promptly.

The system analysis for the "Car vs. Dog Prediction Project" provides a comprehensive overview of the project's objectives, stakeholders, requirements, data flow, technology stack, constraints, and a timeline for execution. This analysis serves as a foundation for the subsequent phases of the project, including design, development, testing, and deployment.

3.1 EXISTING SYSTEM

The existing system for the "Cat vs. Dog Prediction Project" typically consists of conventional image classification methods and potentially early versions of deep learning models. Here's an overview of the existing system:

1. Image Classification Algorithms:

Traditional image classification techniques such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), or decision trees may have been used. Handcrafted feature extraction methods like Histogram of Oriented Gradients (HOG) and Scale-Invariant Feature Transform (SIFT) might have been applied.

2. Preprocessing and Feature Engineering:

Image preprocessing involves tasks like resizing, color normalization, and noise reduction.

Feature engineering may include designing and extracting relevant features from the images manually.

3. Training Data:

The training dataset may consist of labeled cat and dog images, manually collected and annotated.

Data augmentation techniques, such as rotation and flipping, might be used to increase dataset diversity.

4. Model Training:

Machine learning models like SVM or k-NN are trained on the preprocessed and feature-engineered data.

The model's parameters are adjusted to optimize classification performance.

5. Evaluation:

The system's performance is evaluated using traditional classification metrics like accuracy, precision, recall, and F1 score.

Cross-validation or holdout validation may be employed to assess model generalization.

6. User Interface:

An existing system may have a simple graphical user interface for users to upload images and receive predictions.

In some cases, it might be a command-line tool.

7. Limitations:

Limited performance: Conventional methods may struggle with complex image data and variations in pose, lighting, and background.

Manual feature engineering: Extracting handcrafted features can be time-consuming and less effective for tasks with large and diverse datasets.

8. Scalability:

The existing system may lack the scalability and adaptability to accommodate new data or emerging deep learning techniques.

9. Ethical Considerations:

The system might not address ethical concerns, such as biases in predictions or data privacy.

The existing system for the "Cat vs. Dog Prediction Project" likely relies on traditional machine learning techniques and manual feature engineering. However, advancements in deep learning have the potential to significantly enhance the accuracy and robustness of such image classification tasks, making them more adaptable to real-world applications. Therefore, transitioning to deep learning models like CNNs is a common progression in projects of this nature.

3.2 PROPOSED SYSTEM

1. Image Data Collection:

- Collect a diverse and balanced dataset of labeled images containing cats and dogs from various sources, including online repositories and custom data collection efforts.
- Ensure data quality and proper annotation to provide reliable ground truth labels.

2. Data Preprocessing:

- Normalize and standardize the collected images, ensuring consistent resolution, color channels, and format.
- Implement data augmentation techniques to increase dataset diversity and improve the model's generalization.

3. Deep Learning Model Development:

- Configure the neural network with appropriate layers, activation functions, and optimizer settings.

- Consider the option of using transfer learning with pre-trained models, such as VGG, ResNet, or Inception, to leverage learned features and enhance model performance.

4. Model Training:

- Train the deep learning model on the preprocessed and augmented dataset using a powerful GPU or cloud-based infrastructure.
- Monitor training progress, validate the model's performance, and fine-tune hyperparameters as needed.

3.3 FEASIBILITY STUDY

Define the Project Scope:

Clearly outline the objectives and goals of the cat vs. dog prediction project. Determine what you want to achieve, such as predicting whether an image contains a cat or a dog based on machine learning or deep learning models.

Market Research:

Research the demand for such a prediction system. Who is the target audience? Are there existing solutions, and if so, how does your project compare? Understand the potential user base and their needs.

Technical Feasibility:

Assess whether the technology and resources required for the project are available. Consider factors like hardware, software, data availability, and the expertise needed to build and maintain the predictive model.

`Data Availability:

Evaluate the availability and quality of data for training and testing the model. For a cat vs. dog prediction system, you'll need a large dataset of labeled cat and dog images. Ensure that acquiring or creating such a dataset is feasible.

Legal and Ethical Considerations:

Investigate any legal and ethical issues related to using images of animals. Make sure you have the right to use the data and that your project complies with data protection and copyright laws.

Cost Analysis:

Calculate the budget required for the project. Consider expenses related to hardware, software, data acquisition, personnel, and ongoing maintenance.

Timeframe:

Determine the project timeline, including development, testing, and deployment. Consider any constraints or deadlines.

Skills and Expertise:

Assess whether you have the necessary skills or if you need to hire or collaborate with experts in machine learning, computer vision, and data science.

Risks and Mitigations:

Identify potential risks that could impact the project's success. Develop mitigation strategies to address these risks.

Alternative Solutions:

Explore alternative methods or technologies that might achieve the same goal more cost-effectively or efficiently.

Decision:

Based on the information gathered during the feasibility study, make a decision on whether to proceed with the project. Consider factors like market demand, technical feasibility, costs, and potential returns.

Documentation:

Prepare a comprehensive feasibility report that summarizes your findings, recommendations, and the rationale for your decision. This document will serve as a reference and justification for the project.

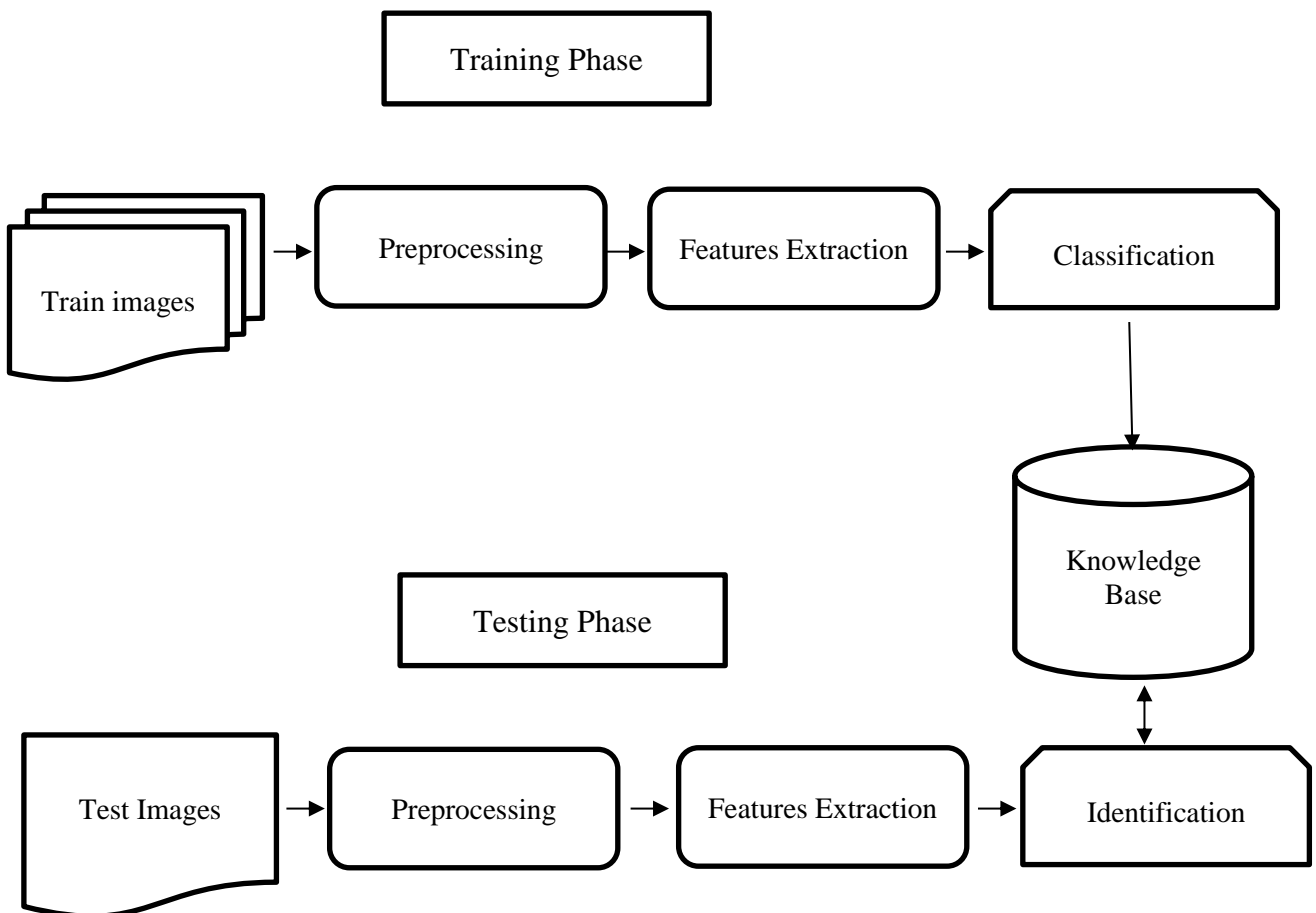
Remember that a feasibility study is a critical step in project planning and can save you time and resources by helping you make an informed decision about whether to move forward with the cat vs. dog prediction project.

CHAPTER 4

SYSTEM DESIGN




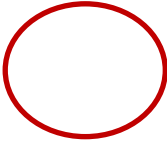
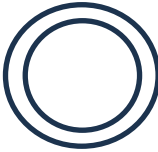


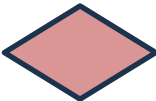
System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It is a critical phase in the development of any complex system, whether it's a software application, hardware system, or a combination of both. System design provides a blueprint for how the system will be built and how it will function.

4.1 SYSTEM FLOW DIAGRAM:

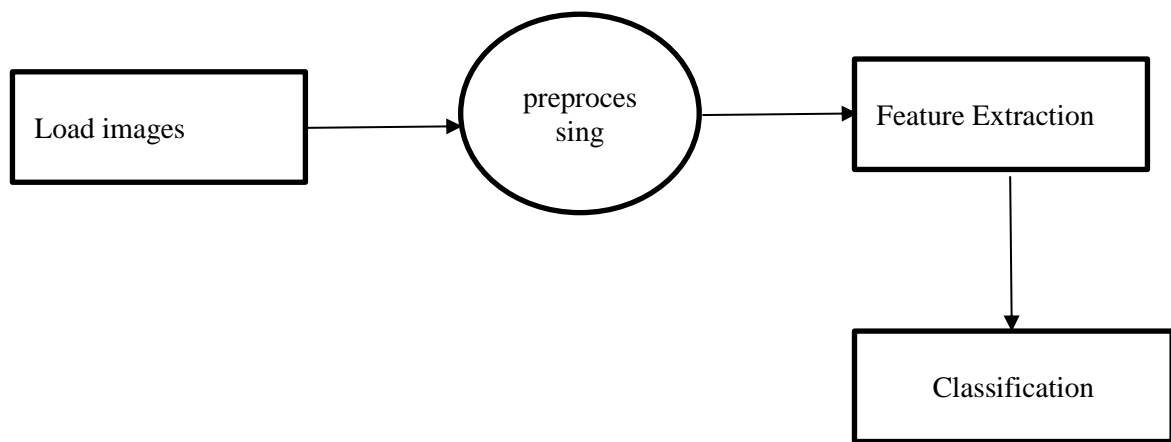


4.2 Flow Diagram:

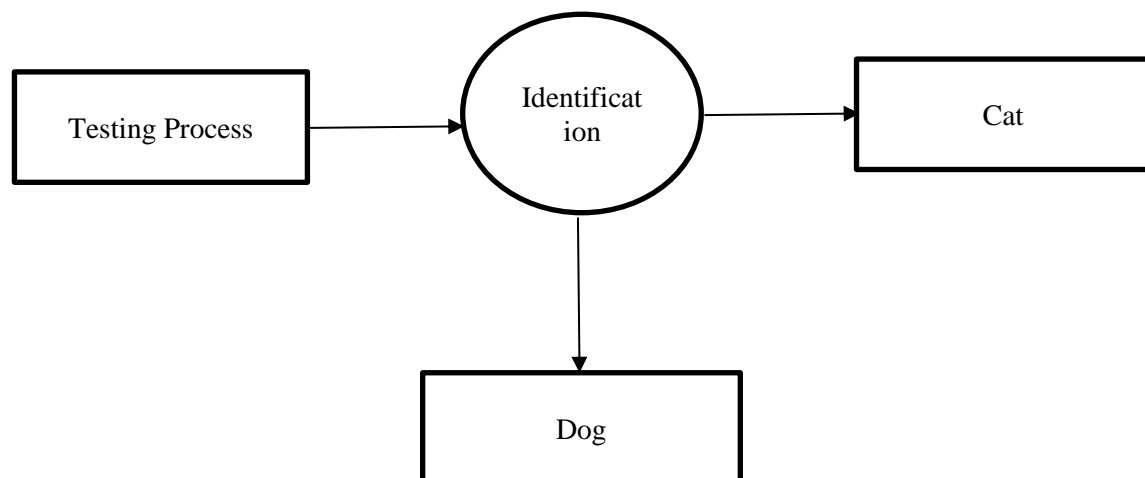
Symbols and Usages

SL.NO	SYMBOLS	USAGES
1		PROCESS
2		EXTERNAL ENTITIES
3		DATA FLOW
4		STATE
5		START STATE
6		STOP STATE
7		DATA STORE
8		ENTITIY RELATIONSHIP

Level 0



Level 1



CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES AND MODULE DESCRIPTION

Implementing a dog vs. cat prediction system involves several steps, including data collection, model development, training, testing, and deployment. Here's a high-level overview of the system implementation process for a dog vs. cat prediction project:

Data Collection:

Gather a large dataset of labeled cat and dog images. You can use publicly available datasets or create your own. Ensure the data is clean and well-organized.

Data Preprocessing:

Clean and preprocess the image data. This may involve resizing, normalization, and data augmentation to improve the quality of the dataset.

Model Selection:

Choose the appropriate model architecture for your image classification task. Convolutional Neural Networks are commonly used for image classification.

Model Development:

Develop the architecture of your neural network. This includes defining layers, activation functions, and optimization algorithms. Make sure to split the data into training, validation, and test sets.

Model Training:

Train the model using the training dataset. Monitor its performance on the validation set to prevent overfitting. Fine-tune hyperparameters as needed.

Model Evaluation:

Evaluate the model's performance on the test dataset using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Make adjustments to improve performance if necessary.

Deployment:

Once you're satisfied with the model's performance, it's time to deploy it for predictions. There are several deployment options, including:

That implementing a dog vs. cat prediction system is an iterative process, and it may require ongoing maintenance and updates to keep the model accurate and the system efficient.

5.2 ALGORITHM AND TECHNIQUES

Algorithm Explanation

Transfer learning is a machine learning technique where a pre-trained model developed for one task is used as the starting point for a model on a second task. It involves taking the knowledge learned from one problem and applying it to a different but related problem. Transfer learning has gained significant popularity in recent years, especially in the field of deep learning, because it allows models to leverage the knowledge acquired from large, complex datasets for a new task, even when the new dataset is relatively small.

Pre-trained Models:

Transfer learning typically starts with a pre-trained model, such as a deep neural network, that was trained on a large dataset for a specific task, like image classification (e.g. ImageNet), natural language processing (e.g. BERT for text understanding), or audio processing. These pre-trained models have already learned useful features and patterns from the source task.

Used Packages:

Sklearn:

- Here, the `train_test_split()` class from `sklearn.model_selection` is used to split our data into train and test sets where feature variables are given as input in the method.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy array and for manipulation purpose.

Matplotlib:

- Matplotlib provides functions to display images in various formats, including grayscale and color images.
- You can use `imshow()` to display images and add titles, labels, and color maps

Mobile Net V2:

- MobileNetV2 is a deep learning architecture designed for image classification and computer vision tasks.
- It's a variant of the original MobileNet architecture, optimized for efficient and lightweight image processing.
- MobileNetV2 has gained popularity for various image classification applications, especially on mobile devices and embedded systems, due to its compact size and good performance.

CHAPTER 6

SYSTEM TESTING

System testing for a cat vs. dog image classification project involves evaluating the performance and functionality of the entire system, including the model, data pipeline. Here's a systematic approach to conducting system testing for your project:

Test Data Preparation:

Ensure that you have a well-organized test dataset with a representative sample of cat and dog images. This dataset should be separate from the training and validation datasets to avoid data leakage.

Preprocessing and Data Pipeline:

Test the data preprocessing and pipeline to ensure that it properly handles image resizing, normalization, and any data augmentation. Verify that the pipeline provides the correct input to the model.

Model Evaluation:

Test the image classification model by using the test dataset. Evaluate its performance using relevant metrics such as accuracy, precision, recall, F1-score, and confusion matrix. Ensure that the model is not overfitting to the test dataset.

Boundary Testing:

Test the system's boundaries by including edge cases, such as images with low quality, unusual angles, or partial views of cats and dogs. Assess how the system handles these scenarios.

Performance Testing:

Evaluate the system's performance by measuring its response time. Test how long it takes to make predictions on various image sizes and quantities. Check for latency and resource usage.

Robustness Testing:

Test the system's robustness by introducing noisy or altered images. Verify that it can handle variations in lighting, backgrounds, and image quality.

Accuracy and Consistency:

Test the model's accuracy and consistency over time. Repeatedly test the system with the same input data to ensure that it consistently provides the same results.

User Acceptance Testing (UAT):

Conduct UAT with target users or stakeholders to gather feedback, assess usability, and ensure that the system meets their requirements and expectations.

Compliance and Legal Testing:

Confirm that the system complies with legal requirements, data protection regulations, and any industry-specific standards.

Performance under Load:

Test how the system performs under heavy load by simulating a large number of concurrent users and image requests. Ensure it remains responsive and functional.

Recovery Testing:

Test how the system recovers from unexpected errors, crashes, or issues, and verify that it can resume normal operation.

Scalability Testing:

Test the system's scalability by gradually increasing the size of the test dataset and checking if it can handle larger workloads.

Document and address any issues or bugs identified during testing, and make necessary adjustments to improve the system's performance and reliability. It's important to perform comprehensive testing to ensure that your cat vs. dog image classification system functions accurately and efficiently in real-world scenarios.

CHAPTER 7

APPENDICES

7.1 SOURCE CODE

```
import os
```

```
#counting the number of files in train folder
```

```
path,dirs,files=next(os.walk('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train'))
```

```
file_count=len(files)
```

```
print('Number of image :',file_count)
```

```
#Printing the name of images
```

```
file_names=os.listdir('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train')
```

```
print(file_names)
```

```
#Importing the dependencies
```

```
import numpy as np
```

```
from PIL import Image
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as mpimg
```

```
from sklearn.model_selection import train_test_split
```

#Displaying dog images

```
img=mpimg.imread('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\dog.2078.jpg')
```

```
imgplot=plt.imshow(img)
```

```
plt.show()
```

#Displaying cat images

```
img=mpimg.imread('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\cat.208.jpg')
```

```
imgplot=plt.imshow(img)
```

```
plt.show()
```

#Find the total number of cat and dog images

```
file_names=os.listdir('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train')
```

```
dog_count=0
```

```
cat_count=0
```

```
for img_file in file_names:
```

```
    name=img_file[0:3]
```

```
    if name=='dog':
```

```
        dog_count +=1
```

```
    else:
```

```
        cat_count +=1
```

```
print("Number of dog images=",dog_count)
```

```
print("Number of cat images=",cat_count)
```

```
#Resizing all the images
```

```
#creating a directory for resized images
```

```
os.mkdir('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\image_resized')
```



```
original_folder='D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\'
```

```
resized_folder='D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\image_resized\\'
```

```
for i in range(2000):
```

```
    filename=os.listdir(original_folder)[i]
```

```
    img_path=original_folder+filename
```

```
    img=Image.open(img_path)
```

```
    img=img.resize((224,224))
```

```
    img=img.convert('RGB')
```

```
    newimgpath=resized_folder+filename
```

```
    img.save(newimgpath)
```

#Displaying resize cat images

```
img=mpimg.imread('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\image_resized\\cat.1.jpg')
```

```
imgplot=plt.imshow(img)
```

```
plt.show()
```

#Displaying resize dog images

```
img=mpimg.imread('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\image_resized\\dog.1358.jpg')
```

```
imgplot=plt.imshow(img)
```

```
plt.show()
```

#Createing Labels for resized images of dogs and Cats

#creating a for loop to assign labels

```
filenames=os.listdir('D:\\python-Project\\Classification using Transfer  
Learning\\train\\train\\image_resized')
```

```
labels=[]
```

```
for i in range(2000):
```

```
    file_name=filenames[i]
```

```
    label=file_name[0:3]
```

```
    if label=='dog':
```

```
labels.append(1)

    else:

        labels.append(0)
```

```
print(filenamees[0:5])

print(len(filenamees))

print(labels[0:5])

print(len(labels))
```

#creating the images of dogs and cats out of 2000 images

```
values,counts=np.unique(labels,return_counts=True)

print(values)

print(counts)
```

#converting all the resized images to numpy array

```
import cv2

import glob

image_directory='D:\\python-Project\\Classification using Transfer
Learning\\train\\train\\image_resized\\'

image_extension=['png','jpg']
```

```
files=[]
```

```
[files.extend(glob.glob(image_directory + '*' + e))for e in image_extension]
```

```
dog_cat_images=np.asarray([cv2.imread(file) for file in files])
```

```
print(dog_cat_images)
```

```
type(dog_cat_images)
```

```
print(dog_cat_images.shape)
```

```
X=dog_cat_images
```

```
Y=np.asarray(labels)
```

```
#Train Test Split
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
#scaling the data
```

```
X_train_scaled=X_train/255
```

```
X_test_scaled=X_test/255
```

```
print(X_train_scaled)
```

#Building the neural network

```
import tensorflow as tf
```

```
import tensorflow_hub as hub
```

```
mobilenet_model='https://tfhub.dev/google/tf2-  
preview/mobilenet_v2/feature_vector/4'
```

```
pretrained_model=hub.KerasLayer(mobilenet_model,input_shape=(224,224,3  
,trainable=False)
```

```
num_of_classes=2
```

```
model=tf.keras.Sequential([  
  
    pretrained_model,  
  
    tf.keras.layers.Dense(num_of_classes)  
  
])
```

```
model.summary()
```

```
model.compile(  
    optimizer='adam',
```

```
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
```

```

metrics=['acc']

)

model.fit(X_train_scaled,Y_train,epochs=5)

score,acc=model.evaluate(X_test_scaled,Y_test)

print('Test Loss=',score)

print('Test Accuracy=',acc)

```

#Predictive System

```

input_image_path=input('path of the image to be predicted: ')

input_image=cv2.imread(input_image_path)

cv2_imshow=(input_image)

input_image_resize=cv2.resize(input_image,(224,224))

input_image_scaled=input_image_resize/255

image_reshaped=np.reshape(input_image_scaled,[1,224,224,3])

input_prediction=model.predict(image_reshaped)

input_pred_label=np.argmax(input_prediction)

if input_pred_label==0:

    print('The image represents a cat')

else:

```

```
print('The image represents a Dog')
```

7.2 SCREENSHOTS

COUNTING THE NUMBER OF FILE IN TRAIN FOLDER

```
import os
#counting the number of files in train folder
path,dirs,files=next(os.walk('D:\\python-Project\\Classification using Transfer Learning\\train\\train'))
file_count=len(files)
print('Number of image :',file_count)
```

✓ 0.1s

Number of image : 2101

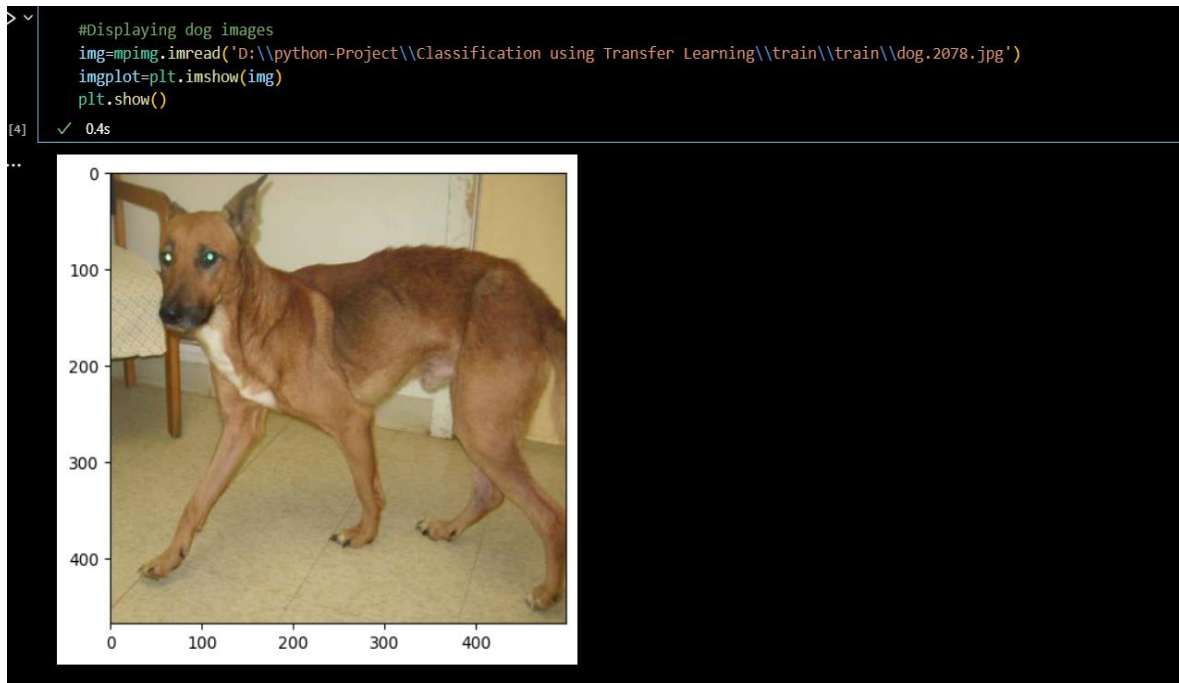
PRINTING THE NAME OF THE IMAGES

```
file_names=os.listdir('D:\\python-Project\\Classification using Transfer Learning\\train\\train')
print(file_names)
```

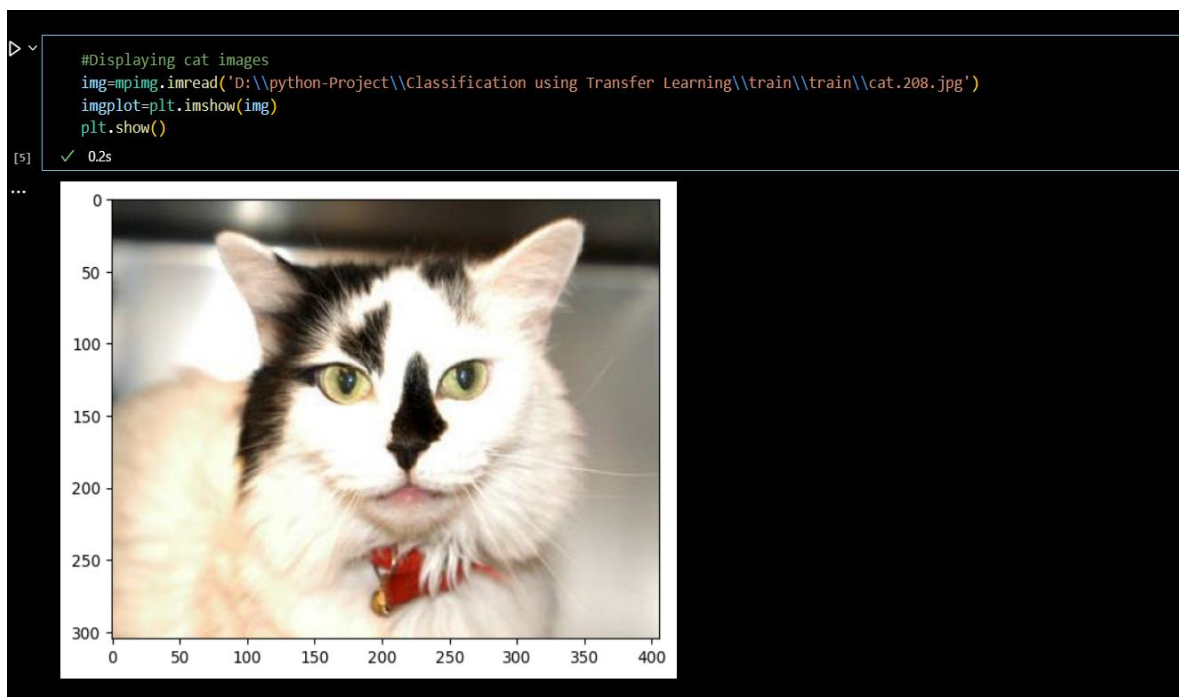
[2] ✓ 0.0s Python

... ['cat.0.jpg', 'cat.1.jpg', 'cat.10.jpg', 'cat.100.jpg', 'cat.101.jpg', 'cat.102.jpg', 'cat.103.jpg', 'cat.104.jpg', 'cat.105.jpg', 'cat.106.jpg', 'cat

DISPLAYING DOG IMAGE



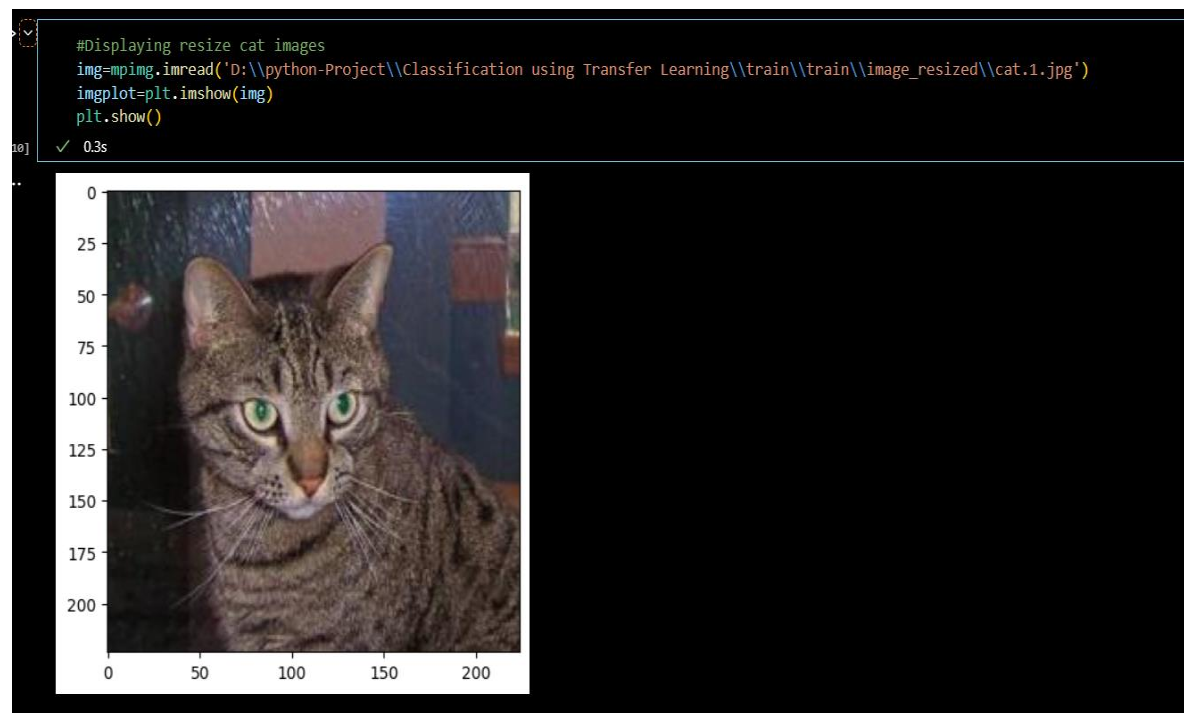
DISPLAYING CAT IMAGE



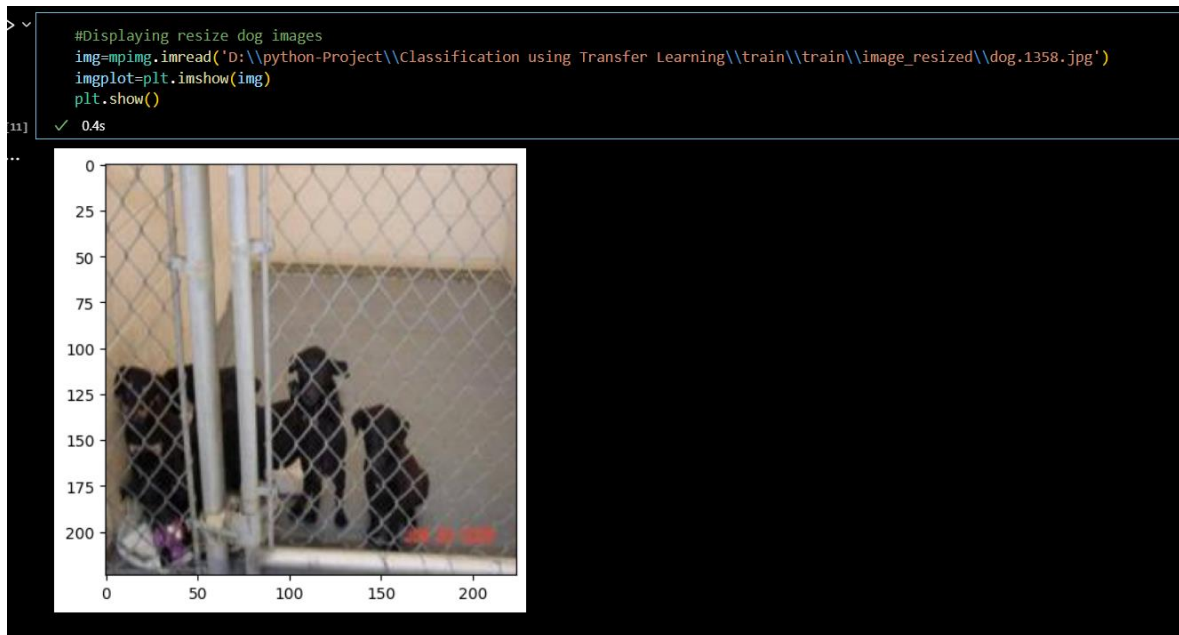
DISPLAYING THE NUMBER OF DOG AND CAT IMAGES

```
Number of dog images= 1101  
Number of cat images= 1001
```

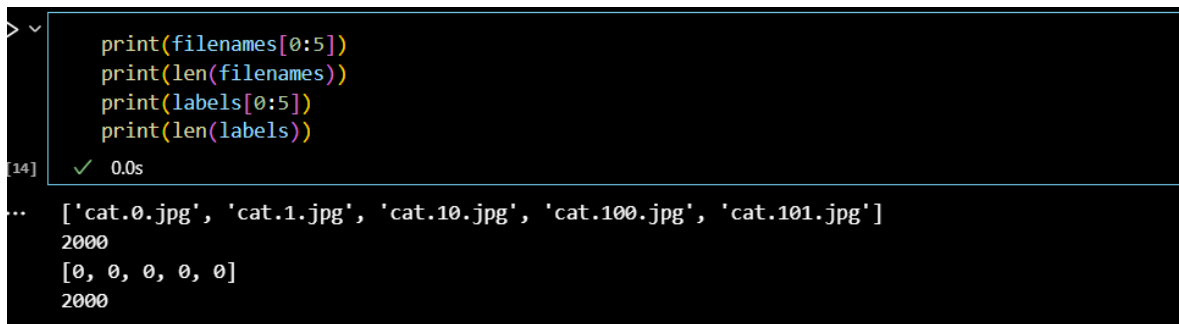
DISPLAYING RESIZE CAT IMAGES



DISPLAYING RESIZE DOG IMAGES



PRINT THE FIENAME,LEN(FILE),LABELS,LEN(LABELS)



CREATING THE IMAGES OF DOGS AND CATS OUT OF 2000 IMAGES

```
> ✓  
#creating the images of dogs and cats out of 2000 images  
values,counts=np.unique(labels,return_counts=True)  
print(values)  
print(counts)  
[15] ✓ 0.0s  
... [0 1]  
      [1000 1000]
```

SCALING THE DATA

```
[16] ✓ 15.9s  
... [[[[ 87 163 205]  
       [ 89 165 207]  
       [ 92 168 210]  
       ...  
       [124 203 246]  
       [122 200 246]  
       [122 200 246]]]  
  
      [[ 87 163 205]  
       [ 89 165 207]  
       [ 92 168 210]  
       ...  
       [124 203 246]  
       [123 201 247]  
       [122 200 246]]]  
  
      [[ 87 163 205]  
       [ 89 165 207]  
       [ 92 168 210]  
       ...  
       [124 204 245]  
       [123 202 245]  
       [123 202 245]]]  
  
      ...  
      ...  
      ...  
      [206 235 232]  
      [224 253 250]  
      [230 255 255]]]]]
```

PRINT THE DOG AND CAT IMAGES SHAPE

```
type(dog_cat_images)
print(dog_cat_images.shape)
```

[17] ✓ 0.0s

... (2000, 224, 224, 3)

TRAIN_TEST_SPLIT DATASET

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
print(X.shape,X_train.shape,X_test.shape)
```

1 ✓ 0.1s

(2000, 224, 224, 3) (1600, 224, 224, 3) (400, 224, 224, 3)

MODEL SEQUENTIAL

```
... Model: "sequential"
```

Layer (type)	Output Shape	Param #
keras_layer (KerasLayer)	(None, 1280)	2257984
dense (Dense)	(None, 2)	2562

=====
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
=====

SCALING THE DATA

```
0] ✓ 7.0s
· [[[[0.77647059 0.79607843 0.78823529]
      [0.77647059 0.79607843 0.78823529]
      [0.77647059 0.79607843 0.78823529]
      ...
      [0.84705882 0.85098039 0.84313725]
      [0.84705882 0.85098039 0.84313725]
      [0.84705882 0.85098039 0.84313725]]]

  [[0.77647059 0.79607843 0.78823529]
   [0.77647059 0.79607843 0.78823529]
   [0.77647059 0.79607843 0.78823529]
   ...
   [0.83529412 0.83921569 0.83137255]
   [0.83529412 0.83921569 0.83137255]
   [0.83529412 0.83921569 0.83137255]]

  [[0.77647059 0.79607843 0.78823529]
   [0.77647059 0.79607843 0.78823529]
   [0.77647059 0.79607843 0.78823529]
   ...
   [0.82745098 0.83137255 0.82352941]
   [0.82352941 0.82745098 0.81960784]
   [0.82352941 0.82745098 0.81960784]]

  ...
  ...
  ...
  [0.29411765 0.24313725 0.23529412]
  [0.29803922 0.24705882 0.23921569]
  [0.30196078 0.25098039 0.24313725]]]]]
```

FIT THE MODULES

```
[25] ✓ 3m 1.7s
... Epoch 1/5
50/50 [=====] - 42s 736ms/step - loss: 0.1736 - acc: 0.9362
Epoch 2/5
50/50 [=====] - 36s 716ms/step - loss: 0.0685 - acc: 0.9762
Epoch 3/5
50/50 [=====] - 34s 672ms/step - loss: 0.0505 - acc: 0.9850
Epoch 4/5
50/50 [=====] - 35s 698ms/step - loss: 0.0402 - acc: 0.9894
Epoch 5/5
50/50 [=====] - 33s 668ms/step - loss: 0.0318 - acc: 0.9925
... <keras.callbacks.History at 0x2166dc70a30>
```

TEST LOSS AND TEST ACCURACY

```
[23]
· 13/13 [=====] - 10s 656ms/step - loss: 0.0760 - acc: 0.9725
Test Loss= 0.07599207758903503
Test Accuracy= 0.9725000262260437
```

FINALE PREDICTION OF IMAGE CLASSIFICATION

```
[25]
· 1/1 [=====] - 0s 65ms/step
The image represents a Dog
```

CHAPTER 8

CONCLUSION

How to use transfer learning for image classification. Transfer learning is flexible, allowing the use of pre-trained models directly as feature extraction preprocessing and integrated into entirely new models. The main goal of this develop a system that can identify images of cats and dogs. The input image will be analyzed and then the output is predicted. The Dogs vs Cats dataset can be downloaded from the Kaggle website. The dataset contains a set of images of cats and dogs. Our main aim here is for the model to learn various distinctive features of cat and dog. Once the training of the model is done it will be able to differentiate images of cat and dog.

we build a deep convolutional neural network for image classification (. Despite of using only a subset of the images and accuracy of 97.10% . If the whole dataset was being used the accuracy would have been even better. This paper shows a classification process using convolutional neural networks which is a deeplearning architecture.

CHAPTER 9

FUTURE ENHANCEMENT

Future enhancements for an image classification project using transfer learning can help improve the model's performance and extend its capabilities.

Multi-Class Classification: Expand the model to predict more than just dogs and cats. Include other common pets, such as rabbits, birds, or fish. This would make the application more versatile and appealing to a broader audience.

Fine-Grained Classification: Improve the model to identify specific dog or cat breeds. This could be valuable for pet owners looking for information about their specific breed.

Real-Time Prediction: Develop a mobile application that allows users to take a photo of a pet and receive an instant prediction. This could include live video analysis, making it more interactive and engaging.

CHAPTER 10

BIBLIOGRAPHY

1. A.Akansu, M.Smith (editors). Subband and Wavelet transform. Design and Applications, Kluwer Academic Publishers, 1996.
2. A Fast Precise Implementation of 8x8 Discrete Cosine Transform Using the Intel® Streaming SIMD Extensions and MMX™ Instructions, Application Note AP922, Intel Corp. Order number 742474, 1999.
3. Fast Algorithms for Median Filtering, Application Note, Intel Corp. Document number 79835, 2001.
4. GB/T 200090.2-2006. China Standard. Information Technology. Coding of Audio-Visual Objects - Part 2: Visual (02/2006).
5. M.Bertalmio, A.L.Bertozzi, G.Sapiro. Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting. Proc. ICCV 2001, pp.1335-1362, 2001.
6. G.Borgefors. Distance Transformations in Digital Images. Computer Vision, Graphics, and Image Processing 34, 1986.
7. J-Y.Bouguet. Pyramidal Implementation of the Lucas-Kanade Feature Tracker. OpenCV Documentation, Microprocessor Research Lab, Intel Corporation, 1999.
8. J. Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence 8(6), 1986.
9. J.Davis and Bobick. The Representation and Recognition of Action Using Temporal Templates. MIT Media Lab Technical Report 402, 1997.
10. J.Davis and G.Bradski. Real-Time Motion Template Gradients Using Intel(R) Computer Vision Library. IEEE ICCV'99 FRAME-RATE WORKSHOP, 1999.
11. N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. INRIA, 2005.