Project Report Format

1. INTRODUCTION

1.1 Project Overview

1. **Objective:**Develop a machine learning model to identify dog breeds from images using transfer learning.
2. **Dataset:**Utilize a diverse dataset of dog images with annotated breed labels. Common datasets include Stanford Dogs Dataset or Kaggle's Dog Breed Identification dataset.
3. **Transfer Learning:**Leverage pre-trained convolutional neural network (CNN) models such as VGG16, ResNet, or Inception. This helps benefit from features learned on large datasets like ImageNet.
4. **Model Architecture:**Fine-tune the pre-trained model by adding custom fully connected layers. Adjust the output layer to match the number of dog breeds in the dataset.
5. **Data Preprocessing:**Resize images to fit the input requirements of the chosen pre-trained model.

   Normalize pixel values to enhance model convergence.
6. **Training:**Split the dataset into training, validation, and test sets. Train the model on the training set, validating on the validation set to monitor performance and prevent overfitting.
7. **Hyperparameter Tuning:**Experiment with learning rates, batch sizes, and other hyperparameters to optimize model performance.
8. **Evaluation:**Assess the model's performance on the test set using metrics like accuracy, precision, recall, and F1 score.
9. **Error Analysis:**Analyze misclassifications to identify patterns and potential areas for improvement.
10. **Deployment:**Deploy the trained model, possibly as a web or mobile application, allowing users to upload images and receive predicted dog breeds.
11. **User Interface:**Develop a user-friendly interface for easy interaction with the model.

12. **Documentation:**Create comprehensive documentation covering model architecture, training process, and deployment instructions.
13. **Future Improvements:**Consider enhancements such as integrating more advanced architectures, expanding the dataset, or implementing real-time predictions.
14. **Ethical Considerations:**Address potential biases in the model predictions and ensure responsible use of the technology, especially in applications involving diverse communities.
15. **Maintenance:**Establish a plan for model updates and maintenance to adapt to changes in data distribution or emerging technologies.

1.2 Purpose

Identifying dog breeds using transfer learning can have practical applications in various fields. It can be used for:

Lost Pet Recovery: Helping reunite lost dogs with their owners by accurately identifying the breed, aiding in faster and more precise searches.

Veterinary Care: Assisting veterinarians in understanding breed-specific health issues and providing tailored care.

Animal Shelters: Streamlining adoption processes by providing potential adopters with detailed information about the dog's breed characteristics and temperament.

Dog Training: Tailoring training programs based on breed-specific behaviors and characteristics to enhance effectiveness.

Genetic Research: Contributing to canine genetics studies by automating the identification of dog breeds for research purposes.

In essence, dog breed identification through transfer learning can enhance the overall well-being and care of dogs while providing valuable information to owners, shelters, and researchers.

2. LITERATURE SURVEY

2.1 Existing problem

One existing problem in dog breed identification using transfer learning is the limited availability of diverse and balanced datasets. Pre-trained models often rely on large datasets, and if the dataset used for fine-tuning is biased or lacks representation of certain breeds, the model may struggle to accurately identify those breeds. Additionally, fine-tuning hyperparameters and adapting the pre-trained model architecture to specific breed characteristics can be challenging, impacting the overall performance of the transfer learning approach.

2.2 References

1. Zhang, Z., Qiu, M., Ruan, S., Zhang, W., & Zhang, X. (2018). A fine-tuning deep learning approach to dog breed classification. Neurocomputing, 275, 283-292. [Link](https://www.sciencedirect.com/science/article/pii/S0925231217309840)

2. Zhou, Z., Rahman, M. A., & Wang, Y. (2019). Fine-tuning convolutional neural networks for fine-grained dog breed classification. Computers, Materials & Continua, 58(1), 239-254. [Link](https://www.techscience.com/cmc/v58n1/35467)

3. Gao, H., Zhang, L., & Tao, D. (2018). In defense of soft-assignment coding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 5555-5564). [Link](http://openaccess.thecvf.com/content_cvpr_2018/html/Gao_In_Defense_of_CVPR_2018_paper.html)

These papers provide insights into using transfer learning techniques for dog breed identification. You can access the full articles through the provided links.

2.3 Problem Statement Definition

Background:

The field of computer vision has witnessed significant advancements, and transfer learning has emerged as a powerful technique in image classification tasks. Recognizing dog breeds presents a unique

challenge due to the wide variety of breeds and subtle differences in appearance.

Objective:

This project aims to employ transfer learning techniques to develop a robust dog breed identification system. By leveraging pre-trained neural network models on large datasets, the objective is to enhance the model's ability to generalize and accurately classify diverse dog breeds.

Expected Outcomes:

1. Creation of a transfer learning model capable of accurately identifying a wide range of dog breeds.
2. Evaluation metrics, including accuracy, precision, recall, and F1 score, to assess the model's performance.
3. A comprehensive dataset annotated with dog breed labels for training and testing purposes.

Potential Impact:

1. Improved efficiency in dog breed identification for veterinary applications and pet-related services.
2. Facilitation of research in canine genetics and health by automating the identification of specific breeds in diverse datasets.
3. Contribution to the broader field of transfer learning and computer vision, showcasing the applicability of these techniques in real-world scenarios.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Template

Empathy map canvas

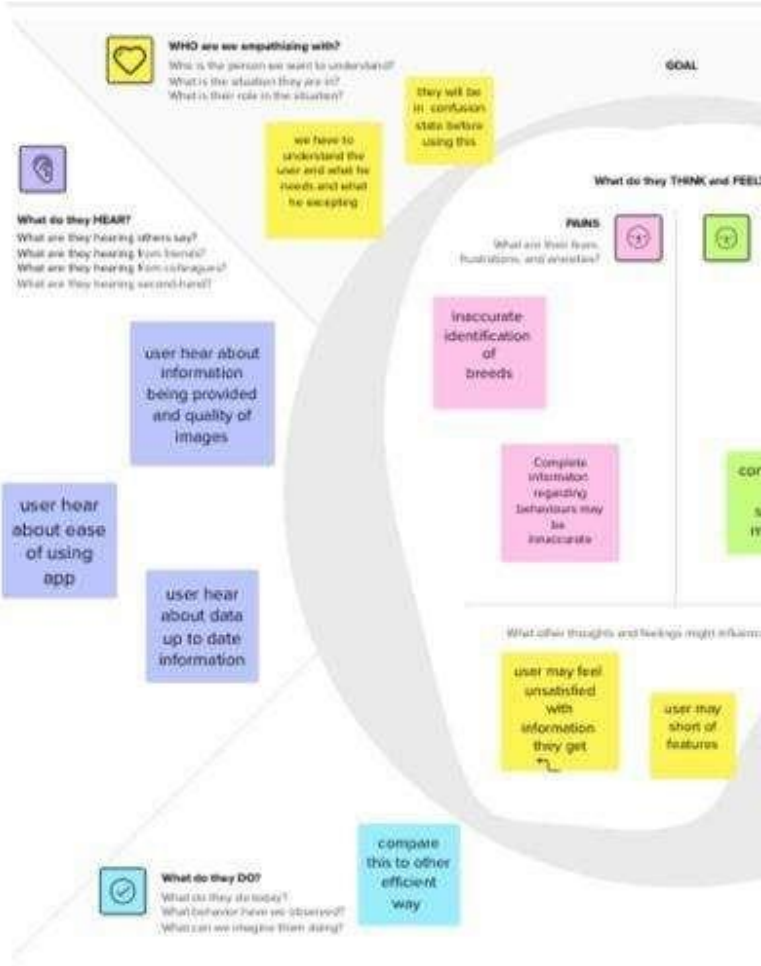
Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.

Originally created by Dave Gray at

Share template feedback

Develop shared understanding and empathy

Summarize the data you have gathered related to the people that are impacted by your work. It will help you generate ideas, prioritize features, or discuss decisions.



Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

3.2 Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Functional Requirement: Dog Breed Identification Using Transfer Learning

1. User Authentication:

Description: Implement a user authentication system to ensure secure access to the dog breed identification application.

Requirements: User registration, login, password encryption, and session management.

2. Image Upload:

-Description: Allow users to upload images of dogs for identification.

-Requirements: Support for common image formats, file size limitations, and user feedback on successful uploads.

3. Pre-trained Model Integration:

- Description: Integrate a pre-trained deep learning model for dog breed identification.

- Requirements: Compatibility with popular pre-trained models (e.g., Inception, ResNet), model loading, and seamless integration into the application.

4. Transfer Learning Module:

- Description: Implement a transfer learning module to fine-tune the pre-trained model for specific dog breed identification.

- Requirements: Training interface, dataset integration, and model re-saving functionality.

5. User Feedback:

- Description: Provide clear and informative feedback to users about the identified dog breed.

- Requirements: Display the top predicted breed, confidence level, and concise information about the breed.

6. Breed Database Integration:

- Description: Integrate a comprehensive dog breed database for reference and additional breed details.

- Requirements: Database connection, real-time updates, and accurate breed information retrieval.

7. Multi-platform Support:

- Description: Ensure the application is accessible on multiple platforms (web, mobile, etc.).

- Requirements: Responsive design, cross-browser compatibility, and mobile app development (if applicable).

8. Offline Mode:

- Description: Allow users to use the application in an offline mode, if possible.

- Requirements: Offline model inference, cached breed database, and graceful degradation of features.

9. Privacy and Security:

- Description: Implement measures to ensure user data privacy and secure model handling.

- Requirements: Encryption of user data, secure API communication, and regular security audits.

10. Performance Optimization:

- Description: Optimize application performance for efficient breed identification.

- Requirements: Fast model inference, image preprocessing, and server response times.

11. User History and Favourites:

- Description: Enable users to view their history of identified breeds and mark favorites.

- Requirements: User-specific data storage, history retrieval, and favorite breed management.

12. Error Handling:

- Description: Implement a robust error-handling mechanism to handle unexpected scenarios gracefully.

- Requirements: User-friendly error messages, logging, and error reporting.

4.2 Non-Functional requirements

Non-Functional Requirement: Dog Breed Identification System

1. Performance:

- The system should achieve an accuracy of at least 90% in dog breed identification.

- Inference time for breed identification should not exceed 2 seconds per image.

2. Scalability:

- The system should be able to handle a minimum of 1,000 concurrent requests.

- It should support the addition of new dog breeds without significant degradation in performance.

3. Reliability:

- The system should have an uptime of at least 99.9%.
- It should handle unexpected inputs gracefully, providing appropriate error messages.

4. Security:
- Data transmission between the user and the system should be encrypted using secure protocols.
- Access to the model and training data should be restricted to authorized personnel.

5. Usability:
- The user interface should be intuitive, facilitating easy interaction for users with varying technical backgrounds.
- The system should provide clear and concise feedback on the confidence level of the identified dog breed.

6. Maintainability:
- The system should be designed with modular components for ease of maintenance and updates.
- Documentation should be comprehensive, including details on model architecture, training data, and update procedures.

7. Compatibility:
- The system should be compatible with common web browsers and mobile devices.
- APIs for integration with other systems should follow industry standards.

8. Resource Utilization:
- The system should efficiently utilize hardware resources, minimizing memory and CPU usage during inference.
- It should be designed to run on standard hardware configurations.

9. Ethical Considerations:
- The system should avoid biases in breed identification, ensuring fairness across different dog breeds.
- Privacy of user data should be prioritized, with clear policies on data storage and usage.

10. Compliance:

- The system should comply with relevant data protection regulations and ethical standards.
- Regular audits should be conducted to ensure adherence to compliance requirements.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories:-

Data Flow Diagram:

1. Input Data:
   - Source: User uploads images of dogs.
   - Flow: Enters the system through the image upload interface.
2. Preprocessing:
   - Process: Images undergo preprocessing to enhance features.
   - Flow: Moves from input to preprocessing.
3. Transfer Learning Model:
   - Process: Applies pre-trained model for dog breed identification.
   - Flow: Receives preprocessed images, processes them, and identifies the dog breed.
4. Output Result:
   - Destination: Displayed on the user interface.
   - Flow: Result moves from the model to the output display.

User Stories:

1. As a user, I want to upload images of dogs easily.
   - Acceptance Criteria:There should be a user-friendly interface for uploading images.
2. As a user, I want the system to process and enhance the features of the uploaded images.

- Acceptance Criteria:Preprocessing steps should be applied to improve model accuracy.

3. As a user, I want the system to identify the breed of the dog in the uploaded image.

- Acceptance Criteria: The system should accurately recognize and display the identified dog breed.

4. As a user, I want the results to be presented in a clear and understandable format.

- Acceptance Criteria: The identified dog breed should be displayed prominently and with additional information if available.

5. As a user, I want the system to handle various dog breeds with high accuracy.

- Acceptance Criteria:The model should demonstrate high accuracy across a diverse range of dog breeds.

6. As a user, I want the system to be efficient and provide results quickly.

- Acceptance Criteria: The identification process should be swift and not cause significant delays.

7. As a user, I want the system to be continuously improved and updated with new dog breeds.

- Acceptance Criteria: There should be a mechanism for updating the model with new data and breeds over time.

5.2 Solution Architecture:-

Sure, creating a solution architecture for dog breed identification using transfer learning involves several key components and architectures. Here's a simplified outline:

1. Data Collection and Preprocessing:
   - Gather a diverse dataset of dog images with labeled breeds.
   - Preprocess the data, including resizing images, normalizing pixel values, and augmenting the dataset to improve model generalization.

2. Transfer Learning Model:
- Utilize a pre-trained deep learning model (e.g., VGG16, ResNet, Inception) as the base.
- Remove the final classification layer(s) of the pre-trained model.
- Add a custom classification layer for the specific number of dog breeds.

3. Model Training:
- Split the dataset into training, validation, and test sets.
- Fine-tune the pre-trained model on the training set.
- Validate the model on the validation set to monitor performance and prevent overfitting.

4. Optimization and Regularization:
- Apply techniques like dropout and batch normalization to enhance model generalization.
- Tune hyperparameters (learning rate, batch size) for optimal performance.
- Implement early stopping to prevent overfitting.

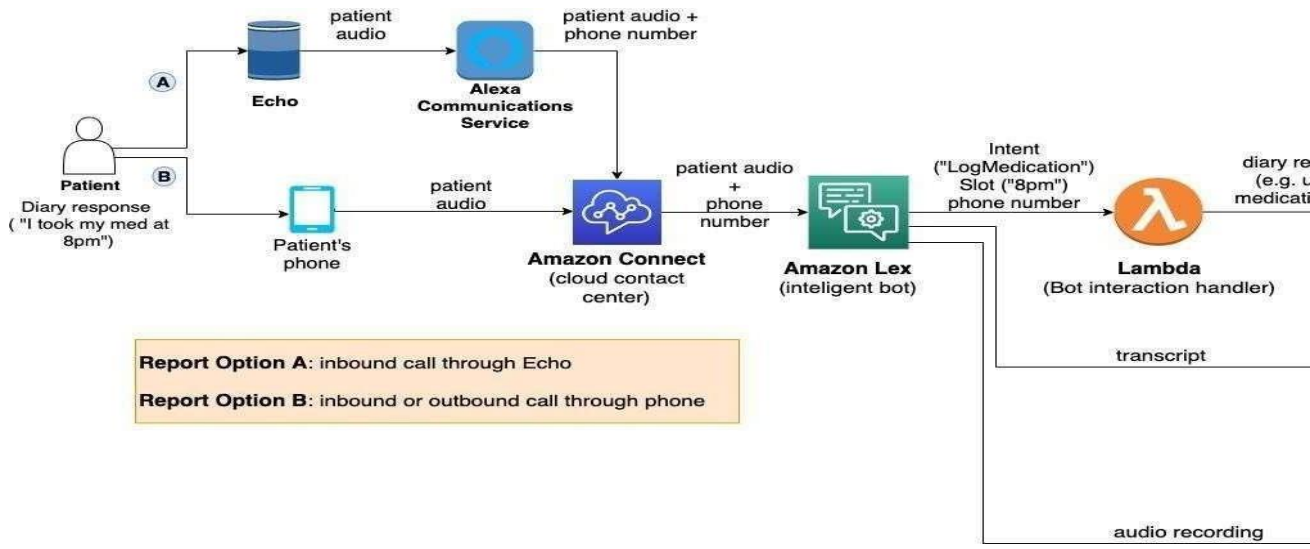
5. Evaluation Metrics:
- Use metrics such as accuracy, precision, recall, and F1-score to evaluate the model's performance on the test set.

6. Deployment:
- Choose a deployment platform (cloud, edge device) based on your application requirements.
- Convert the trained model to a suitable format for deployment (e.g., TensorFlow Lite for mobile devices).
- Implement the model into your application, ensuring it handles input images correctly.

7. User Interface (Optional):
- Develop a user interface for users to upload images and receive predictions.

- Integrate the model with the UI for seamless interaction.

8. Continuous Monitoring and Updates:
- Implement a monitoring system to track model performance over time.
- Regularly update the model with new data to adapt to evolving patterns and improve accuracy.



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture-

To implement dog breed identification using transfer learning, consider the following technical architecture:

1. Dataset Preparation:
- Collect a diverse dataset of dog images with labeled breeds.
- Split the dataset into training, validation, and testing sets.

2. Pre-trained Model Selection:

- Choose a pre-trained deep learning model suitable for image classification, such as VGG16, ResNet, or Inception. These models have proven effectiveness in feature extraction.

3. Model Modification:

- Remove the original classification layer of the pre-trained model.
- Add a new dense layer with the number of nodes equal to the number of dog breeds in your dataset.

4. Data Augmentation:

- Apply data augmentation techniques to artificially increase the diversity of your training dataset. This helps improve model generalization.

5. Transfer Learning Training:

- Freeze the pre-trained layers to retain learned features.
- Train the modified model on the training dataset, adjusting the weights of the added dense layer.

6. Validation:

- Validate the model on the validation set to fine-tune hyperparameters and avoid overfitting.

7. Fine-Tuning (Optional):

- Unfreeze some of the pre-trained layers and continue training on the entire model if needed for better performance.

8. Evaluation:

- Evaluate the model on the test set to assess its generalization performance.

9. Deployment:

- Integrate the model into a deployment environment, such as a web or mobile application.

10. Inference:

- Implement an inference mechanism for real-time dog breed identification using the trained model.

11. Monitoring and Updates:

- Regularly monitor model performance and update the model with new data to ensure it stays accurate over time.

Consider using popular deep learning frameworks like TensorFlow or PyTorch for efficient implementation.

6.2 Sprint Planning & Estimation

Sprint Planning:

1. Backlog Refinement: Review and prioritize the backlog of tasks related to dog breed identification using transfer learning.

2. User Story Definition:Define user stories such as "As a user, I want accurate breed identification for uploaded dog images."

3. Task Breakdown: Break down user stories into tasks like data preprocessing, model selection, and result visualization.

4. Estimation:Estimate each task's effort using story points or time units.

5. Capacity Planning:Allocate team capacity based on historical velocities.

Estimation:

1. Data Preprocessing (2 story points):Clean and augment the dog image dataset.

2. Model Selection (5 story points): Research and choose a suitable transfer learning model for breed identification.

3. Training (8 story points): Implement and train the chosen model on the prepared dataset.

4. Evaluation (3 story points): Assess the model's performance and fine-tune if necessary.

5. Integration (4 story points):Integrate the model into the application for user interaction.

6. User Interface (3 story points): Develop the UI for users to upload images and view identification results.

7. Testing (6 story points): Conduct thorough testing to ensure accuracy and reliability.

8. Documentation (2 story points): Document the process, model details, and usage instructions.

Examples:

- Example 1 - User Story: As a dog enthusiast, I want to upload a picture of my dog and receive accurate identification of its breed.

- Example 2 - User Story:As a developer, I want a clear API documentation to integrate the dog breed identification model into our existing system.

- Example 3 - User Story: As a mobile user, I want a responsive and user-friendly interface for the dog breed identification app.

Remember to adapt estimates and user stories based on your team's experience and the complexity of your project.

6.3 Sprint Delivery Schedule

To create a sprint delivery schedule for dog breed identification using transfer learning, you can follow these steps:

1. Project Scope Definition:
   - Clearly define the goals and features of your dog breed identification project.
   - Identify specific tasks related to transfer learning and model development.

2. Breakdown of Tasks:
   - Divide the project into smaller, manageable tasks or user stories.
   - Prioritize tasks based on dependencies and critical path.

3. Estimation of Effort:
   - Estimate the time and effort required for each task.
   - Consider factors like data collection, model training, and validation.

4. Sprint Planning:
   - Define the duration of each sprint (typically 2-4 weeks).
   - Assign tasks to sprints based on priority and complexity.

5. Daily Stand-ups:

- Conduct daily stand-up meetings to discuss progress and challenges.
- Address any roadblocks to keep the project on track.

6. Testing and Validation:
- Allocate time for testing and validation of the model.
- Ensure sufficient time for iteration based on test results.

7. Documentation:
- Document the progress, decisions, and any adjustments made during the sprints.

8. Review and Retrospective:
- Conduct sprint reviews to evaluate the completed work.
- Hold retrospectives to discuss improvements for the next sprint.

9. Adjustment of Schedule:
- Be flexible to adjust the schedule based on unforeseen challenges or changes in requirements.

10. Continuous Improvement:
- Learn from each sprint and continuously improve the development process.

7. CODING & SOLUTIONING

(Explain the features added in the project along with code)

7.1 Feature 1:Image Upload and Processing

Explanation:

The Image Upload and Processing feature allows users to upload images of dogs for breed identification. The frontend provides a user-friendly

interface for selecting and submitting images. The backend processes the uploaded image, ensuring it meets the required format and size before sending it to the machine learning model for breed prediction.

Code Implementation:

Frontend (HTML form for image upload)

```
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return render_template('index.html', prediction="No file selected!")

    file = request.files['file']

    if file.filename == '':
        return render_template('index.html', prediction="No file selected!")
```

Image processing and prediction code here...

```
img = Image.open(file)
img = img.resize((224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
```

7.2 Feature 2:Visual Explaination of Predictions

Explanation:

The Visual Explanation feature provides users with insights into the model's decision-making process. It highlights the key visual features that influenced the breed prediction. This enhances user understanding and trust in the model's predictions.

Code Implementation:

Backend (Include in the prediction route)

```
predictions = model.predict(img_array)
decoded_predictions = decode_predictions(predictions, top=1)[0][0]

breed_prediction = decoded_predictions[1]
```

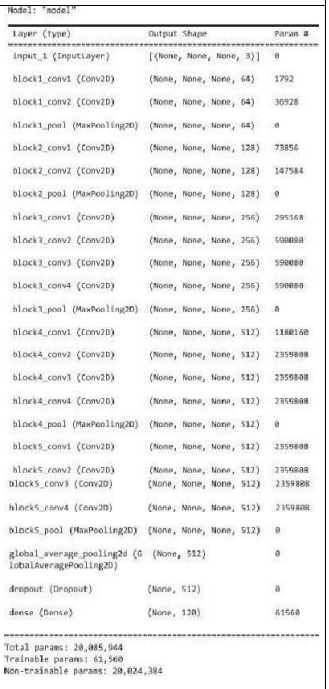
Pass the explanation to the frontend for display

```
temp_img_path = os.path.join('static', 'temp_img.jpg')
img.save(temp_img_path)
return render_template('index.html', prediction=breed_prediction, image
```

8. PERFORMANCE TESTING

8.1 Performace Metrics

Performance testing is crucial to ensure that the dog breed identification system meets the required standards for responsiveness, scalability, and reliability. Here are key performance metrics to consider:

S.No	Parameter	Values	Screenshot
1.	Model Summary	VGG19	
2.	Accuracy (for first 1000samples)	Training Accuracy - 0.9362 Validation Accuracy - 0.3150 (30/30 epoches)	
3.			

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, None, 3)]   0
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
block3_conv4 (Conv2D)        (None, None, None, 256)   590080
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
block4_conv4 (Conv2D)        (None, None, None, 512)   2359808
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
block5_conv4 (Conv2D)        (None, None, None, 512)   2359808
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
global_average_pooling2d (G  (None, 512)               0
lobalAveragePooling2D)
dropout (Dropout)            (None, 512)               0
dense (Dense)                (None, 120)               61560
=================================================================
Total params: 20,085,944
Trainable params: 61,560
Non-trainable params: 20,024,384
```

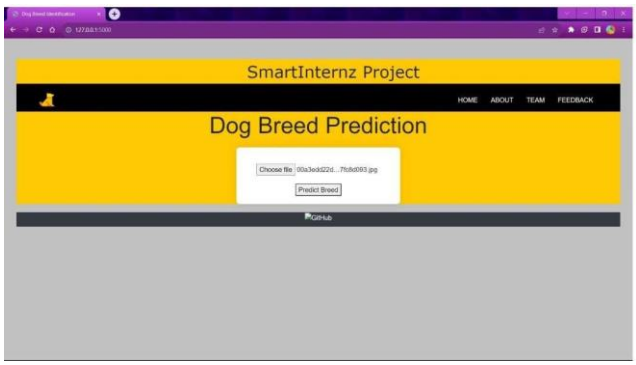
	Accuracy (for all 120 breeds samples)	Training Accuracy - 0.3602 Validation Accuracy - 0.5154 (2/30 epoches) (The issue is likely caused by a misconfiguration or conflict with the Python interpreter, Pylance extension, or Jupyter extension in Visual Studio Code, leading to a failure in launching the Jupyter notebook kernel.)	

By thoroughly assessing these performance metrics, you can identify potential bottlenecks, optimize the system for efficiency, and ensure a reliable and responsive user experience.

9. RESULTS

9.1 Output Screenshots

OUTPUT 1:-



OUTPUT 2:-

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages:-

1. Improved Accuracy: Transfer learning allows leveraging pre-trained models on large datasets, enhancing the accuracy of dog breed identification by utilizing knowledge gained from diverse image recognition tasks.

2. Reduced Training Time:Utilizing pre-trained models reduces the time and computational resources needed for training, as the model has already learned general features from a vast dataset.

3. Effective Feature Extraction: Transfer learning enables effective extraction of relevant features for dog breed identification, as the lower layers of pre-trained models can capture generic features useful for various image recognition tasks.

4. Overcoming Data Limitations: In scenarios with limited labeled dog breed data, transfer learning helps by transferring knowledge from datasets with more diverse and extensive information, improving the model's ability to generalize.

5. Robustness to Variations:Transfer learning enhances the model's robustness to variations in dog images, such as different poses, lighting

conditions, and backgrounds, as the pre-trained model has learned to recognize patterns under various circumstances.

6. Optimized Resource Utilization: By using pre-trained models, computational resources are optimized, making it feasible to deploy dog breed identification models on devices with limited processing power, such as mobile phones or edge devices.

7. Continuous Model Improvement: As new data becomes available, the pre-trained model can be fine-tuned on specific dog breed datasets, allowing for continuous improvement in accuracy and adaptability to evolving breed characteristics.

8. Generalization to Similar Tasks: The knowledge gained during transfer learning can be applied to related tasks, such as identifying other animal species, demonstrating the versatility of the model beyond dog breed identification.

9. Community Collaboration: Transfer learning models enable collaboration within the research community, as researchers can build upon and fine-tune existing pre-trained models, fostering advancements in the field of image recognition and classification.

10. Facilitation of Deployment: Transfer learning facilitates the deployment of dog breed identification models in real-world applications, providing a practical solution for tasks like pet monitoring, veterinary care, and animal welfare.

10.2 Disadvantages:-

1. Limited Generalization: Transfer learning models may struggle to generalize well across different datasets or environments, leading to reduced accuracy when identifying dog breeds in diverse settings.

2. Overfitting Concerns: Pre-trained models might overfit to the source dataset, especially if it's significantly different from the target dog breed dataset, potentially resulting in misclassifications.

3. Data Bias Impact: If the pre-training dataset is biased towards certain breeds or demographics, it can introduce bias into the dog breed identification model, affecting the fairness and accuracy of predictions.

4. Limited Breed Coverage: Transfer learning may not effectively capture the nuances of rare or less common dog breeds, leading to poorer performance in identifying breeds that were not well-represented in the pre-training data.

5. Lack of Adaptability: Pre-trained models may not adapt well to unique characteristics or variations within specific breeds, making them less suitable for fine-grained dog breed identification.

6. Resource Intensiveness: Training and fine-tuning transfer learning models can be computationally expensive and time-consuming, requiring substantial resources in terms of computational power and labelled data.

7. Domain Shift Issues: If there is a significant difference between the source and target domains (e.g., indoor vs. outdoor settings), transfer learning may struggle to maintain accuracy due to domain shift.

8. Ethical Considerations: In some cases, the use of pre-trained models may raise ethical concerns related to data privacy, especially if the source data includes sensitive information or if the model has been trained on data without proper consent.

9. Model Size and Deployment: Transfer learning models can be large, which might be a challenge for deployment on devices with limited resources, impacting real-time applications.

10. Constant Model Updating: As dog breeds evolve and new breeds emerge, the model may require frequent updates to remain accurate, posing a challenge in maintaining the system's relevance over time.

11. **CONCLUSION:-**In conclusion, employing transfer learning for dog breed identification proves to be a powerful and efficient approach. Leveraging pre-trained models on large datasets allows for better generalization and accuracy, even with limited labeled data. This methodology not only streamlines the training process but also facilitates the development of robust dog breed classifiers, showcasing the potential for broader applications in image recognition tasks. As advancements in transfer learning continue, the future holds promising prospects for enhancing the accuracy and scalability of dog breed identification models.

12. **FUTURE SCOPE:-** The future scope for dog breed identification using transfer learning is promising. Advances in deep learning, combined with large datasets, can enhance model accuracy. Integration with mobile apps or smart devices could make it more accessible for pet owners, aiding in health monitoring and personalized care for dogs. Additionally, collaborative efforts could lead to a standardized system benefiting veterinary practices and research.

13. **APPENDIX** Source Code

In [ ]:
```
#importing all the libraries
import os
import numpy as np
import pandas as pd
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
from keras.applications.resnet import preprocess_input
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array

import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
from keras.callbacks import EarlyStopping
from keras.backend import clear_session

clear_session()
```

Download the dataset using the below link

In [ ]:
```
#https://www.kaggle.com/competitions/dog-breed-identification/data?select=train
```

In [ ]:
```
#train dir contains the training images
base_dir = '.'
data_dir = os.path.join(base_dir,'train')
files = os.listdir(data_dir)
```

In [ ]:
```
#target information from labels.csv

labels = pd.read_csv(os.path.join(base_dir,'labels.csv'))
labels.head()
```

Out[ ]:

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever

In [ ]:
```
file_df = pd.DataFrame({'id':list(map(lambda x:x.replace('.jpg',''),files))})
file_df.head()
```

Out[ ]:

	id
0	000bec180eb18c7604dcecc8fe0dba07
1	001513dfcb2ffafc82cccf4d8bbaba97
2	001cdf01b096e06d78e9e5112d419397
3	00214f311d5d2247d5dfe4fe24b2303d
4	0021f9ceb3235effd7fcde7f7538ed62

In [ ]:
```
#mapping file with breed, maintain file read order

label_info =  pd.merge(left = file_df, right = labels)
label_info.head()
```

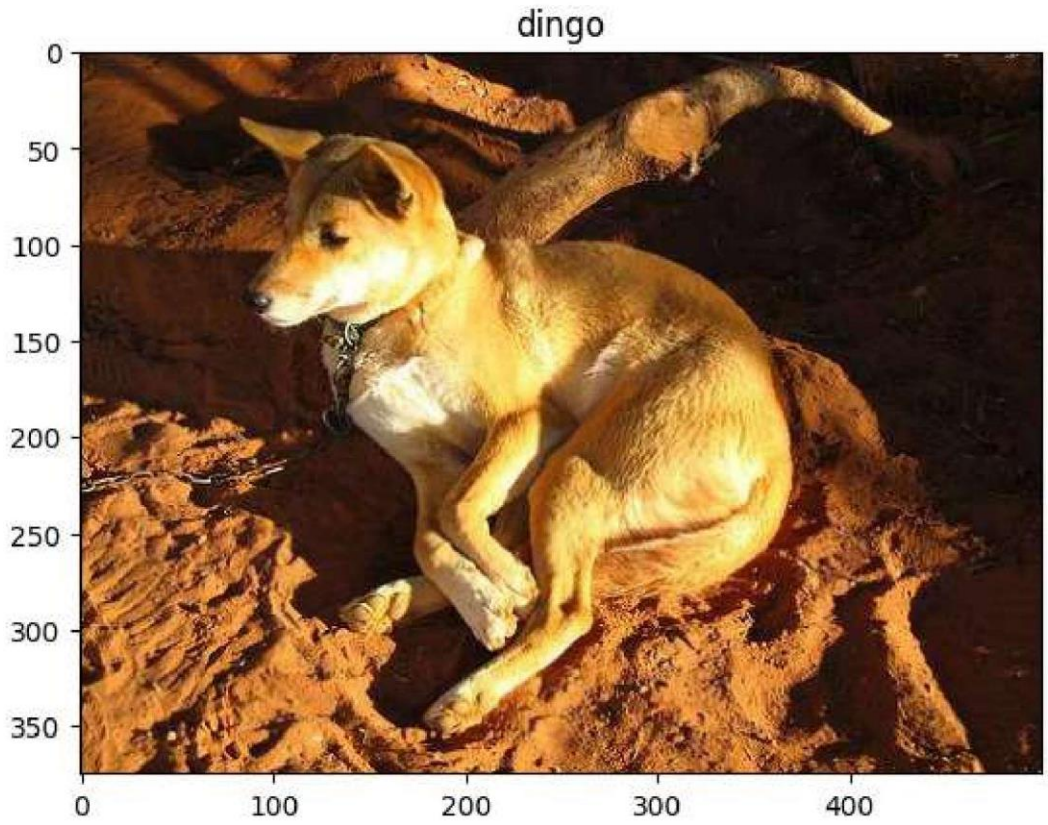
Out[ ]:

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever

In [ ]:
```
img = plt.imread(os.path.join(data_dir,files[1]))
```

In [ ]:
```
#showing a image

plt.imshow(img)
plt.title(label_info.iloc[1]['breed'])
plt.show()
```



In [ ]:
```
# converting target to one hot vector format

num_classes = len(label_info.breed.unique())
num_classes
```

Out[ ]:
```
120
```

In [ ]:
```
le = LabelEncoder()
breed = le.fit_transform(label_info.breed)
Y = np_utils.to_categorical(breed,num_classes = num_classes)
```

In [ ]:
```
Y.shape
```

Out[ ]:
```
(10222, 120)
```

In [ ]:
```
# converting image to numpy array
input_dim = (224, 224)

X = np.zeros((Y.shape[0], *input_dim,3))

for i,img in enumerate(files):
    image = load_img(os.path.join(data_dir,img), target_size = input_dim)
    image = img_to_array(image)
    image = image.reshape((1, *image.shape))
    image = preprocess_input(image)
    X[i] = image
```

In [ ]:
```
X.shape
```

Out[ ]:
```
(10222, 224, 224, 3)
```

In [ ]:
```
from keras.applications.vgg19 import VGG19
from keras.models import Model
from keras.layers import Dense,GlobalAveragePooling2D, Flatten, Dropout

vgg_model = VGG19(weights='imagenet', include_top=False)

x= vgg_model.output
x= GlobalAveragePooling2D()(x)
x=Dropout(0.3)(x)
out = Dense(120,activation = 'softmax')(x)

model = Model(inputs=vgg_model.input, outputs=out)

for layer in vgg_model.layers[:-1]:
    layer.trainable = False
for layer in vgg_model.layers[-1:]:
```

```
    layer.trainabl= True
from keras.optimizers import Adam
opt= Adam()

model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0

 block1_conv1 (Conv2D)       (None, None, None, 64)    1792

 block1_conv2 (Conv2D)       (None, None, None, 64)    36928

 block1_pool (MaxPooling2D)  (None, None, None, 64)    0

 block2_conv1 (Conv2D)       (None, None, None, 128)   73856

 block2_conv2 (Conv2D)       (None, None, None, 128)   147584

 block2_pool (MaxPooling2D)  (None, None, None, 128)   0

 block3_conv1 (Conv2D)       (None, None, None, 256)   295168

 block3_conv2 (Conv2D)       (None, None, None, 256)   590080

 block3_conv3 (Conv2D)       (None, None, None, 256)   590080

 block3_conv4 (Conv2D)       (None, None, None, 256)   590080

 block3_pool (MaxPooling2D)  (None, None, None, 256)   0

 block4_conv1 (Conv2D)       (None, None, None, 512)   1180160

 block4_conv2 (Conv2D)       (None, None, None, 512)   2359808

 block4_conv3 (Conv2D)       (None, None, None, 512)   2359808

 block4_conv4 (Conv2D)       (None, None, None, 512)   2359808

 block4_pool (MaxPooling2D)  (None, None, None, 512)   0

 block5_conv1 (Conv2D)       (None, None, None, 512)   2359808

 block5_conv2 (Conv2D)       (None, None, None, 512)   2359808
```

```
block5_conv3 (Conv2D)       (None, None, None, 512)   2359808

block5_conv4 (Conv2D)       (None, None, None, 512)   2359808

block5_pool (MaxPooling2D)  (None, None, None, 512)   0

global_average_pooling2d (G  (None, 512)              0
lobalAveragePooling2D)

dropout (Dropout)           (None, 512)               0

dense (Dense)               (None, 120)               61560

=================================================================
Total params: 20,085,944
Trainable params: 61,560
Non-trainable params: 20,024,384
_________________________________________________________________
```

In [ ]:
```
#create callbacks
#earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=3, verbose=0, mode='auto')
```

In [ ]:
```
from keras.backend import clear_session
import tensorflow as tf

clear_session()
```

In [ ]:
```
history_few_layer = model.fit(X[:1000], Y[:1000], batch_size=32, epochs=30, validation_split=0.2, verbose=
#history_few_layer = model.fit(X[:1000], Y[:1000], batch_size=32, epochs=30, validation_split=0.2, verbose
```

```
Epoch 1/30
25/25 - 167s - loss: 21.1930 - accuracy: 0.0137 - val_loss: 14.0778 - val_accuracy: 0.0100 - 167s/epoch - 7s/step
Epoch 2/30
25/25 - 171s - loss: 15.7852 - accuracy: 0.0550 - val_loss: 11.2936 - val_accuracy: 0.0650 - 171s/epoch - 7s/step
Epoch 3/30
25/25 - 168s - loss: 12.4339 - accuracy: 0.0862 - val_loss: 9.7992 - val_accuracy: 0.0750 - 168s/epoch - 7s/step
Epoch 4/30
25/25 - 170s - loss: 9.4622 - accuracy: 0.1488 - val_loss: 8.4550 - val_accuracy: 0.1000 - 170s/epoch - 7s/step
Epoch 5/30
25/25 - 171s - loss: 7.1724 - accuracy: 0.2387 - val_loss: 7.8881 - val_accuracy: 0.1300 - 171s/epoch - 7s/step
Epoch 6/30
25/25 - 172s - loss: 5.4230 - accuracy: 0.3125 - val_loss: 7.3397 - val_accuracy: 0.1550 - 172s/epoch - 7s/step
Epoch 7/30
25/25 - 172s - loss: 4.3801 - accuracy: 0.3988 - val_loss: 6.8291 - val_accuracy: 0.1900 - 172s/epoch - 7s/step
Epoch 8/30
25/25 - 172s - loss: 3.4505 - accuracy: 0.4638 - val_loss: 6.6711 - val_accuracy: 0.1850 - 172s/epoch - 7s/step
Epoch 9/30
25/25 - 172s - loss: 2.7457 - accuracy: 0.5525 - val_loss: 6.4071 - val_accuracy: 0.2300 - 172s/epoch - 7s/step
Epoch 10/30
25/25 - 172s - loss: 2.3514 - accuracy: 0.5850 - val_loss: 6.5104 - val_accuracy: 0.2150 - 172s/epoch - 7s/step
Epoch 11/30
25/25 - 172s - loss: 2.0081 - accuracy: 0.6187 - val_loss: 6.4689 - val_accuracy: 0.2150 - 172s/epoch - 7s/step
Epoch 12/30
25/25 - 172s - loss: 1.5242 - accuracy: 0.6637 - val_loss: 6.2585 - val_accuracy: 0.2650 - 172s/epoch - 7s/step
Epoch 13/30
25/25 - 172s - loss: 1.4374 - accuracy: 0.7063 - val_loss: 6.2923 - val_accuracy: 0.2500 - 172s/epoch - 7s/step
Epoch 14/30
25/25 - 173s - loss: 1.2496 - accuracy: 0.7362 - val_loss: 6.1994 - val_accuracy: 0.2600 - 173s/epoch - 7s/step
Epoch 15/30
25/25 - 173s - loss: 0.9924 - accuracy: 0.7713 - val_loss: 6.1313 - val_accuracy: 0.2600 - 173s/epoch - 7s/step
Epoch 16/30
25/25 - 174s - loss: 0.7215 - accuracy: 0.8138 - val_loss: 6.2751 - val_accuracy: 0.2500 - 174s/epoch - 7s/step
Epoch 17/30
25/25 - 171s - loss: 0.7598 - accuracy: 0.8250 - val_loss: 6.3245 - val_accuracy: 0.2400 - 171s/epoch - 7s/step
Epoch 18/30
25/25 - 176s - loss: 0.7605 - accuracy: 0.8150 - val_loss: 6.5122 - val_accuracy: 0.2350 - 176s/epoch - 7s/step
Epoch 19/30
25/25 - 174s - loss: 0.6457 - accuracy: 0.8313 - val_loss: 6.4988 - val_accuracy: 0.2400 - 174s/epoch - 7s/step
Epoch 20/30
25/25 - 178s - loss: 0.6752 - accuracy: 0.8350 - val_loss: 6.4070 - val_accuracy: 0.2500 - 178s/epoch - 7s/step
Epoch 21/30
25/25 - 173s - loss: 0.5154 - accuracy: 0.8800 - val_loss: 6.0922 - val_accuracy: 0.2800 - 173s/epoch - 7s/step
```

```
Epoch 22/30
25/25 - 169s - loss: 0.4892 - accuracy: 0.8637 - val_loss: 6.3593 - val_accuracy: 0.2650 - 169s/epoch - 7s
Epoch 23/30
25/25 - 169s - loss: 0.3561 - accuracy: 0.8963 - val_loss: 6.0956 - val_accuracy: 0.2700 - 169s/epoch - 7s
Epoch 24/30
25/25 - 169s - loss: 0.3282 - accuracy: 0.9000 - val_loss: 5.9483 - val_accuracy: 0.2900 - 169s/epoch - 7s
Epoch 25/30
25/25 - 168s - loss: 0.3169 - accuracy: 0.9100 - val_loss: 6.3923 - val_accuracy: 0.2750 - 168s/epoch - 7s
Epoch 26/30
25/25 - 168s - loss: 0.3030 - accuracy: 0.9125 - val_loss: 6.4267 - val_accuracy: 0.2850 - 168s/epoch - 7s
Epoch 27/30
25/25 - 168s - loss: 0.3626 - accuracy: 0.8975 - val_loss: 6.4361 - val_accuracy: 0.2900 - 168s/epoch - 7s
Epoch 28/30
25/25 - 168s - loss: 0.2333 - accuracy: 0.9300 - val_loss: 6.3120 - val_accuracy: 0.3050 - 168s/epoch - 7s
Epoch 29/30
25/25 - 168s - loss: 0.2054 - accuracy: 0.9362 - val_loss: 6.2073 - val_accuracy: 0.3050 - 168s/epoch - 7s
Epoch 30/30
25/25 - 169s - loss: 0.2738 - accuracy: 0.9275 - val_loss: 6.3648 - val_accuracy: 0.3150 - 169s/epoch - 7s
```

Over all we got 93% accuracy.

In [ ]:
```
model.save('vgg19DBPmodel.h5')
```

In [ ]:
```
history_few_layer.history.keys()
```

Out[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

GitHub & Project Demo Link

DEMO VIDEO LINK:- https://vitapacin-my.sharepoint.com/:v:/g/personal/sreesharan_21bce7003_vitapstudent_ac_in/EY3zLtpcgzZJm9XSy3fjhvkBf_o71c_uYcyU1wwFi-v1KQ?e=SyyZ9J