Project Design Phase-I Solution Architecture

Date	23/10/2023
Team ID	Team-592462
Project Name	Project
Maximum Marks	4 Marks

Solution Architecture:

The solution to the problem of counterfeit logo proliferation involves the implementation of a Convolutional Neural Network (CNN) using the VGG-19 architecture for image classification. This robust model is integrated into a user-friendly web application built with the Flask framework, allowing users to easily upload logo images for verification. The CNN, equipped with deep learning capabilities, rapidly distinguishes between authentic and counterfeit logos, offering real-time classification results. This solution enhances brand protection, rebuilds consumer trust, and empowers informed purchasing decisions by ensuring the swift and accurate detection of fake logos.

1. Data Collection and Preprocessing:

- Gather a labeled dataset of real and fake logos.
- Preprocess the images, resizing them to a consistent size (e.g., 224x224 pixels) and normalizing pixel values.

2. Model Development:

- Implement the VGG-19 architecture for logo classification.
- Fine-tune the model using the collected dataset.
- Split the dataset into training, validation, and test sets.

3. Model Training:

- Train the VGG-19 model on the training dataset, using a suitable optimizer (e.g., Adam) and loss function (e.g., binary cross-entropy).
- Use the validation dataset to monitor model performance and prevent overfitting.

4. Model Evaluation:

- Evaluate the trained model on the test dataset, measuring metrics like accuracy, precision, recall, F1 score, and AUC-ROC.

5. Flask Web Application:

- Develop a Flask-based web application to provide a user-friendly interface for logo detection.
- Create a frontend for users to upload images for classification.
- Implement backend routes to handle image uploads, processing, and classification.

6. Model Deployment:

- Deploy the trained model within the Flask application using a framework like Flask-RESTful.
- Set up a web server to host the Flask application (e.g., using Gunicorn).
- Use Nginx or another web server as a reverse proxy for routing requests to the Flask app.

7. User Interface:

- Design a user-friendly interface that allows users to upload images.
- Implement feedback mechanisms to display classification results (real or fake) to users.

8. Security:

- Implement security measures to prevent malicious uploads or attacks.
- Secure the deployed Flask application using HTTPS for data encryption.

9. Continuous Improvement:

- Set up a pipeline for model retraining with new data periodically to improve accuracy and stay updated with counterfeit techniques.

Solution Architecture Diagram: