GitHub Repo: https://github.com/Bhavya227/Toothbrush-Anomaly-Detection

Streamlit App: https://q22czehz6zq4zlkntvajaa.streamlit.app/

Toothbrush Anomaly Detection – Project Report

Project Overview

This project focuses on building an AI-powered application to detect anomalies in toothbrush images using a pre-trained deep learning model. It aims to support quality inspection processes in manufacturing or packaging industries by classifying images of toothbrushes as either *"Good"* or *"Anomaly"*. The application offers a real-time interface using Streamlit where users can upload an image or use their camera to check product quality.

Model Building

- The model was built using **Google's Teachable Machine**, a web-based platform for training machine learning models without coding.
- The model is a **binary classifier** trained to identify:
    - **Good Toothbrush**
    - **Bad Toothbrush**
- **Dataset**: Custom-labelled images were collected and uploaded to train the model for both classes.
- **Training Configuration**:
    - **Epochs**: 90
    - **Batch Size**: 16
    - **Learning Rate**: 0.0001
- The model was trained using a **Convolutional Neural Network (CNN)** architecture provided by Teachable Machine.
- The final trained model was exported in **TensorFlow Keras (.h5)** format.
- The exported model was integrated into a Python-based **Streamlit app** for real-time predictions and user interaction.

Dataset Source

The dataset used to train this model was manually collected and labelled. It consisted of high-quality images representing:

Source: https://www.mvtec.com/company/research/datasets/mvtec-ad/downloads

- Intact toothbrushes with no defects (Good)

- Toothbrushes with visible defects such as broken bristles, bent handles, or dirt (Bad)

Images were captured in various lighting and angle conditions to improve generalization during training.

Streamlit App Features

The application provides a clean and responsive user interface with the following features:

- **File Upload**: Users can upload toothbrush images for inspection.
- **Camera Input**: Users can capture real-time images using their device camera.
- **Live Predictions**: The app predicts and displays class (Good/Anomaly) along with confidence scores.
- **Styled UI**: Uses HTML + Streamlit features for a professional, user-friendly design.
- **Model Info & Description**: Sidebar includes background and purpose of the app.

Technologies Used

- **Python 3.8 (64-bit)**
- **TensorFlow & Keras** – For model loading and inference
- **Streamlit** – For creating the interactive frontend
- **NumPy** – For image processing
- **Pillow (PIL)** – To handle image transformations
- **Teachable Machine** – For model training

Bonus Implementation: Live Camera Input

One of the key features of this application is **live camera input**. This allows the user to click an image using their device's webcam, which is then automatically fed to the model for real-time anomaly detection. This functionality boosts accessibility and simulates real-time inspection environments commonly used in automated quality control.

Project File Structure

Toothbrush_Anomaly_Detector/

├— app.py # Main Streamlit app

├— keras_model.h5 # Trained model file

├— labels.txt # Class labels file

├— requirements.txt # Dependency list

```
├— README.md            # (Optional) Project readme
```

Conclusion

This project successfully demonstrates how deep learning and web technologies can be combined to build a practical and efficient anomaly detection system for quality control. The use of Teachable Machine simplified the training process, while Streamlit provided an elegant interface to interact with the model. The system is lightweight, easy to deploy, and supports both file upload and live camera input, making it versatile for industrial or educational use. With potential integration into production lines, the solution can enhance product reliability and reduce manual inspection time. Future improvements may include auto-labeling from video feeds, mobile compatibility, and training on more diverse datasets to improve robustness.