COMPUTER VISION BASED PRODUCT RECOMMENDER

B.Tech CSE - V Sem - Software Engineering
CS300

VERSION 2 REPORT

NAME	ROLL NO.	PHONE NO.	EMAIL
Rajath C Aralikatti	181CO241	7829158425	rajath.181co241@nitk.edu.in
Sangeeth S V	181CO246	7907932994	sangeeth.181co246@nitk.edu.in

ABSTRACT

Most E-commerce search engines are still largely dependent on keyword matching and the user's knowledge base to find the product that fits most into the customer’s requirement. This is inefficient in the sense that the description (keywords used) can vary a lot from the buyer’s side to the seller’s side. In this paper, we propose adding another layer to the search criteria. This smarter search engine would basically capture an image as the input and try to classify the image into a product description. This can also make searching for a product much faster and easier. Therefore, there is room for improvement in the search process and for making the customer experience smoother. The implementation proposed is a mobile application with a fast and interactive UI that classifies an image into its corresponding product class, using machine learning, which is then used as a search query in Amazon to list the products. To build such an application that has machine learning at its core we need to incorporate a development process/methodology that is able to have a focus on the aspects of ML domain that make it fundamentally different from other software application domains.

KEYWORDS

Mobile application, Software Development Process, Machine Learning (ML) & Artificial Intelligence(AI), Computer Vision & Image Processing, Convolutional Neural Networks (CNN)

INTRODUCTION

The E-commerce system is fast-evolving and online shopping is rapidly becoming an unavoidable part of our daily lives. The latest challenge that online shoppers face is the sheer amount of digital information. This explosive growth creates an information overload, which in turn, inhibits timely access to items of interest on the Internet. Herein lies the need for a recommendation system. Almost every E-commerce website has its own implementation of a recommendation system based on available data such as recent searches, prior purchases, etc. The aim of such a system is to give the customer an efficient and more personalized experience. However, most recommendation systems are text-based and usually rely on keyword matching systems and the user's knowledge base. Moreover, the text-based description of a product can vary a lot from the buyer’s side to the seller’s side. With the rapid development in computer vision owing to the improvements made by neural networks these recent years, we can now make a shift from traditional word-based searching methods to searching methods involving visual similarity [5].

Similar to the way machine learning has impacted how recommendation systems are being built, advances in machine learning have also stimulated widespread interest in incorporating AI capabilities into software and services in other domains. To be able to effectively build such applications new development processes must be employed as traditional development processes cannot be used owing to the many differences we encounter upon incorporating machine learning into an application. These differences arise due to 1) building, monitoring and versioning being more complex than other kinds of software engineering, 2) the modularizing of internal AI components being difficult when incorporating multiple AI techniques for a single task. So to build ML based applications in a smooth way the right choice of a software development process becomes a task of utmost importance. In this paper we also explore how we incorporated existing software processes by modifying them to suit the development of our application based on the literature available for software engineering methodologies for AI.

Our product is a mobile application with a fast and interactive UI that classifies an image into its corresponding product class, using machine learning, which is then used as a search query in Amazon to list the products. We achieve this computer vision task of mapping the image of a product to its product class by using MobileNetV2 as the architecture for the machine learning classification model. We run the model on device and hence the computational resources of the device becomes an important consideration. MobileNetV2 is able to retain similar accuracies as larger models while minimizing the number of operations making it the apt choice.

In machine learning the performance of a model is strongly tied to the kind of data it is trained on. Hence it is always good to have as much data as possible. This makes data collection both during the initial stages and user feedback stages very important. Our application is also built with making the data collection during the user feedback stage an important focus. We collect the image captured and the user-suggested product class as feedback from the user when our application makes a wrong recommendation. We discuss also in our paper how we may use this data to retrain our model to improve its performance.

LITERATURE REVIEW

Saleema Amershi et al. [1] gives a description of a nine-stage workflow for integrating machine learning into application and platform development. A set of best software engineering practices for building applications and platforms relying on machine learning is also described here. We have used the practices described here while implementing our own ML-based application.

Luyang Chen et al. [2] presents a smart search engine for online shopping. An implementation of the image search functionality is discussed.

Mark Sandler et al. [3] describes a new mobile architecture MobileNetV2 that improves the performance of mobile models on multiple tasks and benchmarks. MobileNetV1 introduced the concept of depthwise separable convolutions which reduced the inference time by reducing the total number of multiply-add (M-add) operations by a factor of $\sim k^2$ (where k is the convolution filter size). In most cases, $k=3$ and hence we get a speedup of 8-9 times with little reduction in accuracy of the model. MobileNetV2 introduces the concept of residual blocks and linear activations on top of depthwise separable convolution. Residual blocks allow the free flow of gradients throughout the model which allows us to train extremely deep models with high classification accuracy. A linear activation is necessary before the residual operation in order to avoid loss of information before projecting to a lower dimensional space (ReLU activation cannot be used; it is always zero for negative inputs).

Daniel Kang et al. [4] describes the abstraction of model assertions for monitoring and continuously improving ML models and how such assertions can integrate into the ML development, and its implementation. Model assertions and uncertainty estimates are used in every step of the ML pipeline - from collecting data, labelling it, training the model and during deployment/inference. The authors propose a method to utilize active learning to improve the performance of the model. The model assertions help identify examples which can be used for active learning. The authors also propose a type of model assertion called a consistency assertion

- which is used to automatically generate weakly labelled data to further train the model to improve its performance. The model assertion abstraction defined here is applicable to any general ML system, and is not restricted to our use-case of recommending products from input images.

Sean Bell et al. [5] talks about the visual similarity between objects and the impact it has on the product design. They present a crowdsourced pipeline to match in-situ images and their corresponding product images. They also illustrate how to use this data, using convolutional neural networks, in image search applications like finding a product and finding visually similar products across categories. Our application also seeks to identify product classes from captured images using CNN.

Tom Diethe et al. [6] describes a reference architecture for self-maintaining systems that can learn continually, as data arrives. In environments where there is a need to update / train our model with new data, we need architectures that adapt to shifting data distributions, cope with outliers and retrain when necessary. This represents continual AutoML or Automatically Adaptive Machine Learning. The conclusions drawn in [6] are useful for the development of the feedback mechanism in our application.

MAIN TEXT

Scope of the Work:

The implementation chosen for our product is a mobile application for the very simple reason that it will be the most convenient for the user. The cross-platform mobile application will be able to classify and recommend products based on the image captured by the user. From the image data, the application determines what product is to be searched for. Upon successfully processing the image and producing the text-based classification of the image, the app is redirected to a list of recommended products on Amazon using the Google CustomSearch API. If the recommendation was wrong or only partially correct, then, a mechanism is available to collect the feedback which is then stored in firebase to refine the machine learning model at a later time.

The application UI will be developed using flutter as it is a free and open source platform independent framework. The underlying image classification unit will use MobileNetV2 for processing the image and producing the recommendation. For some images, it is not enough to just classify it. For example, properly searching for a book requires identifying the title of the book also. In these cases, an OCR text recognition algorithm can also be applied to improve our app. This functionality is not currently included in the design of the app as of now. After the specified requirements have been met, this functionality may be included if possible.

The main deliverable of this project is a cross-platform mobile app that would perform the following tasks:

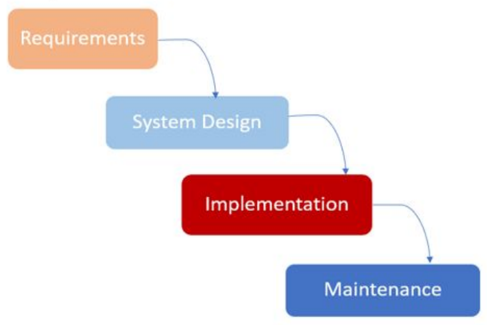
❏ Captures the image and feeds it to the image processing unit.

❏ The image classifier uses the tensorflow model run on the device to process the image and output the text-based classification of the captured image.

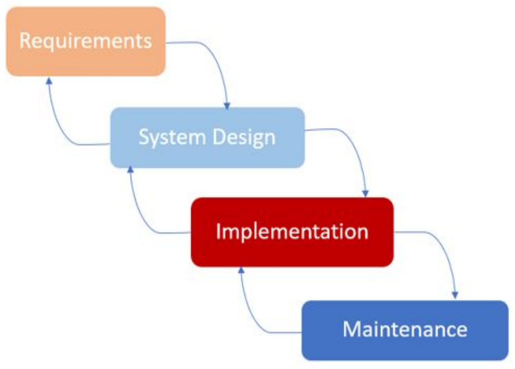
- ❏ Feedback mechanism which collects the feedback and stores it which can then be used to improve/refine the model at a later stage.

<u>Software Engineering Applications:</u>

The software development life cycle proposed is the waterfall model with backflow. This improves upon the traditional waterfall model by incorporating a backflow and provision for getting customer feedback. We took influence from the agile model wherein, at the end of every sprint (dev cycle) the product owner will validate the delivery and provide feedback. Collecting feedback from the product owner at frequent intervals throughout the product development cycle will enable us to ensure that whatever is being delivered is in accordance with the client's requirements. This will also help in reducing the amount of wasted effort due to gaps in requirements gathering. However, our software has well-understood, unambiguous and stable requirements. For such a scenario, the agile methodology is not apt. However, feedback from the client is a necessary step in the development cycle of any software. For this reason, a backflow is incorporated into the waterfall model for smooth development.



Waterfall Model



Waterfall Model with Backflow

The feedback unit of our implementation of the mobile app lies in the maintenance part of the waterfall model specified above. The feedback unit seeks to apply the concepts of continual learning [6] to gather data and use it to refine the model. When the image processing is done, the application will search the recommended product on Amazon and return the relevant results. If the recommendation was wrong or only partially correct, a mechanism is implemented to collect the user-suggested class. Then, in keeping with the requirements specified, the feedback (the captured image and the user-suggested class) is uploaded to the image database.

However this system runs the risk of using invalid/wrong data to train the image classifier. For example, a user can suggest that the picture of a table is a chair. This feedback data should also undergo some checks before being added to the training data. In machine learning, the performance of the model is very strongly tied to the kind of data it is trained on. So, to build ML based applications which continuously learn, the right choice of a software development process becomes a task of utmost importance.

This is how existing software processes, related to machine learning and artificial intelligence [1], have been incorporated by modifying them to suit the development of our application.

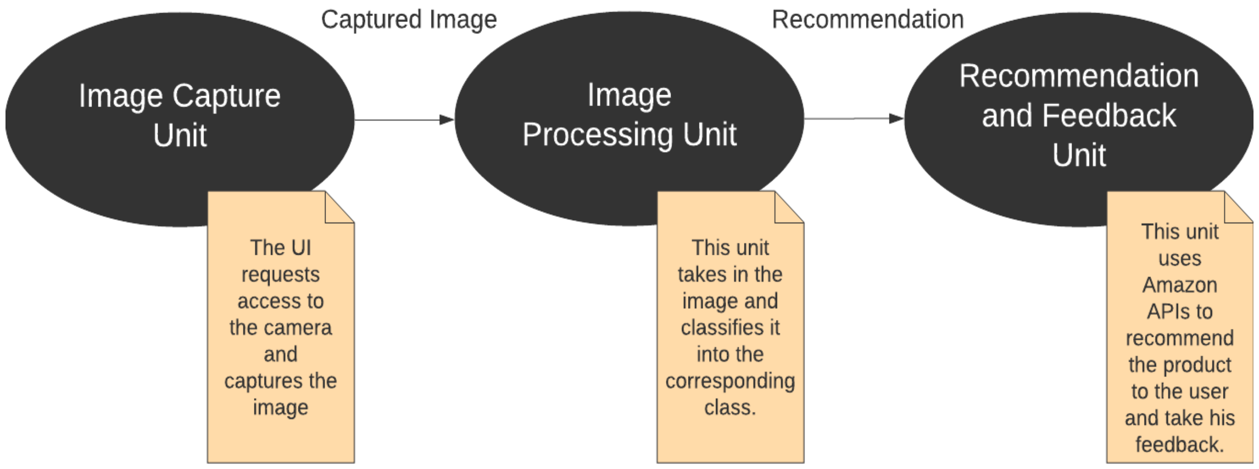
Model:

Model Driven Design (MDD) is described as the separation of business decisions from the software oriented decisions by making use of the interoperability of formal models. Basically, we use *models* (may be conceptual, logical, physical, etc) to describe the functionality of the *solution,* independent of any software-oriented decisions.

We will use a visual (conceptual) model to describe the working and operations performed by the image based product recommender. This model is described according to the requirements specification and does not depend on any implementation details. This model is later used to implement the proposed mobile application.

The mobile app itself can be divided into three main divisions:

- ❏ Image Capture: This unit deals with capturing the image and feeding it as input to the image processing unit. This requires access to the mobile camera so that the image can be captured.
- ❏ Image Processing Unit: The image processing unit deals with computing on the image and providing text based keywords for the closest matching classifications.
- ❏ Recommendation and Feedback Unit: This classification acts as a keyword for an API-based search on Amazon (or any other shopping website). On completion of this recommendation, user’s feedback will be requested which will then be used to improve our model.


Captured Image
Recommendation
Image Capture Unit
Image Processing Unit
Recommendation and Feedback Unit
The UI requests access to the camera and captures the image
This unit takes in the image and classifies it into the corresponding class.
This unit uses Amazon APIs to recommend the product to the user and take his feedback.

This model is independent of the programming language that will eventually be used for the implementation of the mobile application. This conceptual model describes the sequence of the operations performed in the application.

<u>**Working:**</u>

The image based product recommender does the following tasks,

- ❑ It uses the camera of the user's mobile device (after asking user's permission of course) in order to get the image needed.
- ❑ It then captures the image when the user taps on the capture button and the user is then redirected to a confirmation screen where the user can either continue with the currently captured image or go back to the camera to retake the image.
- ❑ Once the user is satisfied with the captured image, it is then passed as input to the image classifier.
- ❑ The image classifier makes use of the MobileNetV2 architecture trained over the ImageNet dataset as the ML model. With a softmax operation at the final output layer in the ML model we get a probability distribution for the product class over the 1000 supported classes of the ImageNet dataset.
- ❑ We then return the output classes with the highest probabilities as a list of possible product classes.
- ❑ The user can go through this list of classifications and click on the closest match. This triggers the opening of a webview within the application and redirects the user to the Amazon product list view.

- When the user returns to our application, an alert immediately pops up and asks the user if the experience was satisfactory and the recommendation was correct. If the user says that the recommendation is correct, then the app redirects to the home screen.
- If the user says that the recommendation was wrong, then he is directed to another screen with a text field which allows the user to enter a suggestion. This suggestion along with the image and the recommendation our app provided, is stored in a database maintained in firebase.
- Then, after the feedback is collected, the user is then redirected to the home screen.

A flowchart showcasing the major components of the image based product recommendation system is shown below:

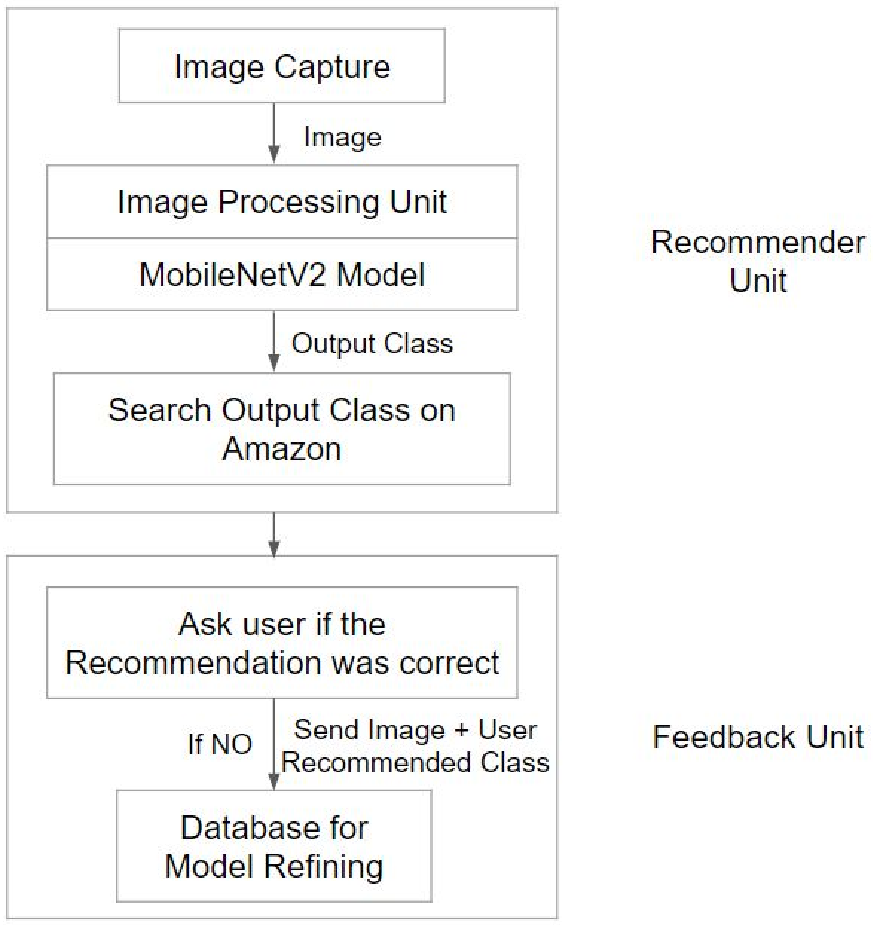


Image Capture
Image
Image Processing Unit
MobileNetV2 Model
Output Class
Search Output Class on Amazon
Recommender Unit
Ask user if the Recommendation was correct
If NO
Send Image + User Recommended Class
Database for Model Refining
Feedback Unit

Platforms and Languages Used:

The image based product recommender requires the following open-source/readily-available platforms/language support.

- ❏ Operating System : Android 8+ / IOS
- ❏ Frontend: Flutter/ Dart constitutes the codebase and SDK used to develop the application.
- ❏ Backend: Firebase for the efficient storage and retrieval of user feedback data.
- ❏ Image Processing Unit: Uses MobileNetV2 architecture [3] for the ML model with python and tensorflow.

Why flutter?

Flutter is a free and open source mobile UI framework from Google that provides a fast and expressive way for developers to build native apps on both IOS and Android. It facilitates fast development, expressive and flexible UI and very high performance apps with minimal effort on the developer's part. Because it is platform independent, it can be used by both android and IOS developers with the same codebase and it automatically gets converted to the corresponding codebase: swift or objective C on iOS and kotlin or java on Android. This eases the effort for the developer during the initial development phase by requiring him to maintain only one codebase for both platforms. This also decreases the time and thus, the cost of updates and maintenance of the app. Moreover, flutter is completely free and open source, which allows for the incorporation of many packages and features that the open source community has developed. The framework also gets regular updates from Google's flutter team making it one of the most popular frameworks for app development. For these reasons, flutter has been chosen as the implementation framework for the image based product recommender.

Why MobileNetV2?

In the implementation proposed, the image processing unit is run on the device and hence neural networks that require high computational resources cannot be used. MobileNetV2 pushes the state of the art techniques for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy, making it an appropriate choice for the machine learning architecture used.

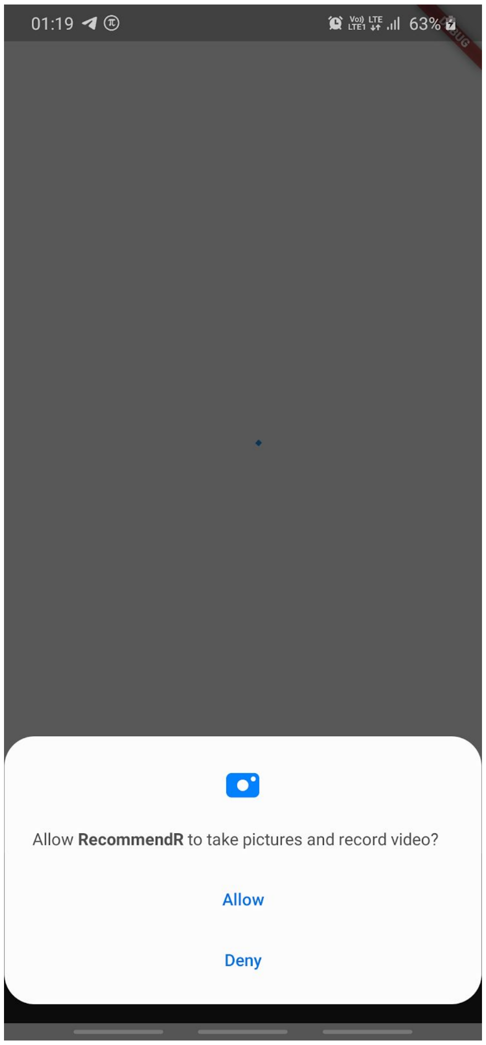
Contributions to Software Engineering:

The image based product recommender is a mobile application where the various stages of the machine learning workflow can be clearly identified. One of the reasons for this is because in our application data collection during the user feedback stage is a major focus. After sufficient data is collected in the feedback database maintained in firebase, we can also look into how it can be best used to improve the ML model. This phase would involve verifying the data collected using appropriate content filters and model retraining, testing and redeployment. Because of these reasons our application is a good example of a general ML based application. Examining how our application is structured and built can help readers better understand some of the stages involved in a ML workflow and help them build other simple ML based applications.

<u>Results:</u>

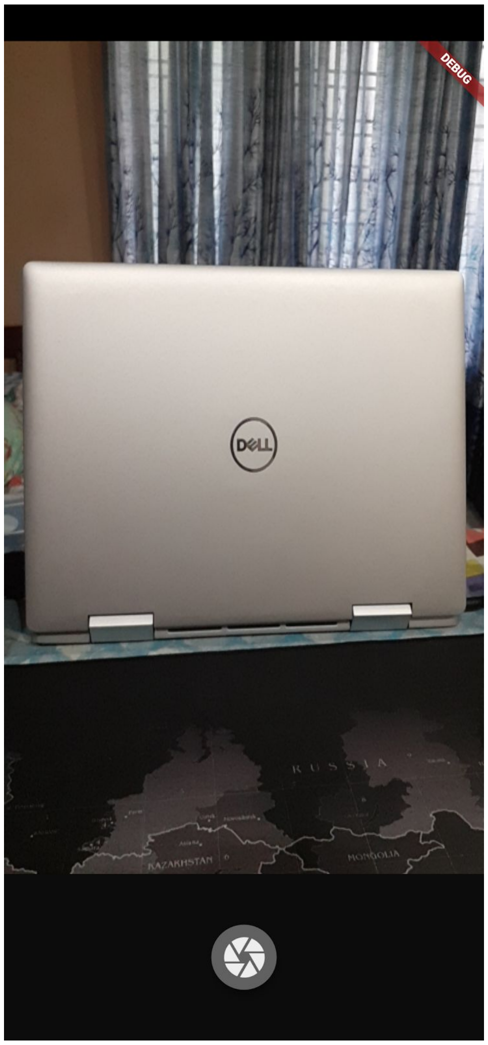
The cross-platform mobile application has been developed using flutter and dart with the tensorflow model added using the tflite package. The screenshots depicting the working of the application have been attached below.

1. <u>Camera Access:</u> The application requires access to the camera for the proper functioning. So, during installation, permission to use the camera is explicitly asked.

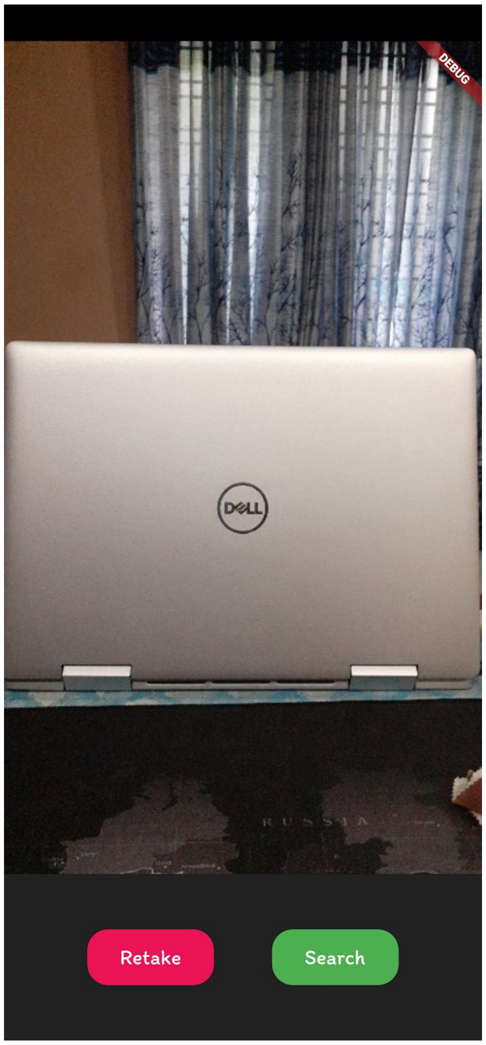


Allow **RecommendR** to take pictures and record video?

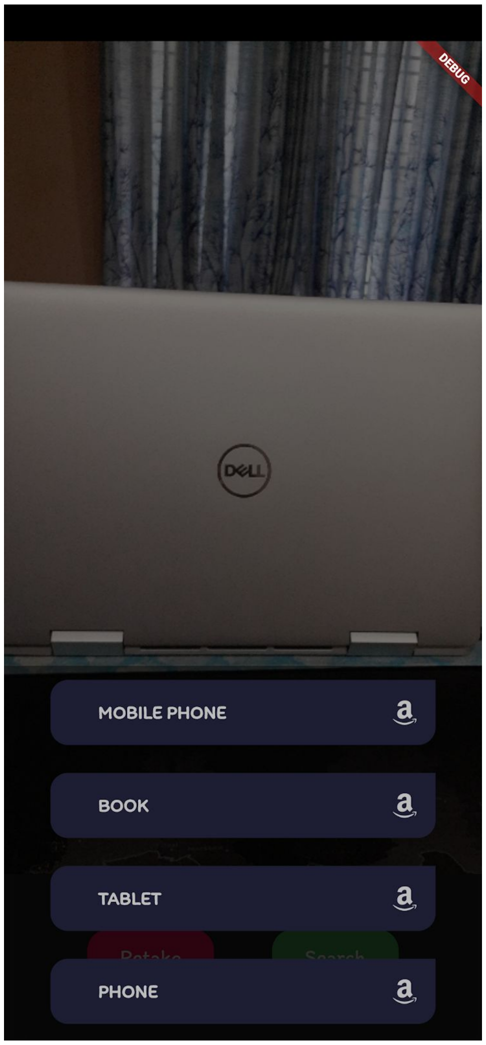
2. Image Capture Screen: This is the home screen where the app shows the camera view and provides a button, which when pressed, will capture the current camera view and create an image file and store it in a temporary location so that it can be used later whenever necessary.



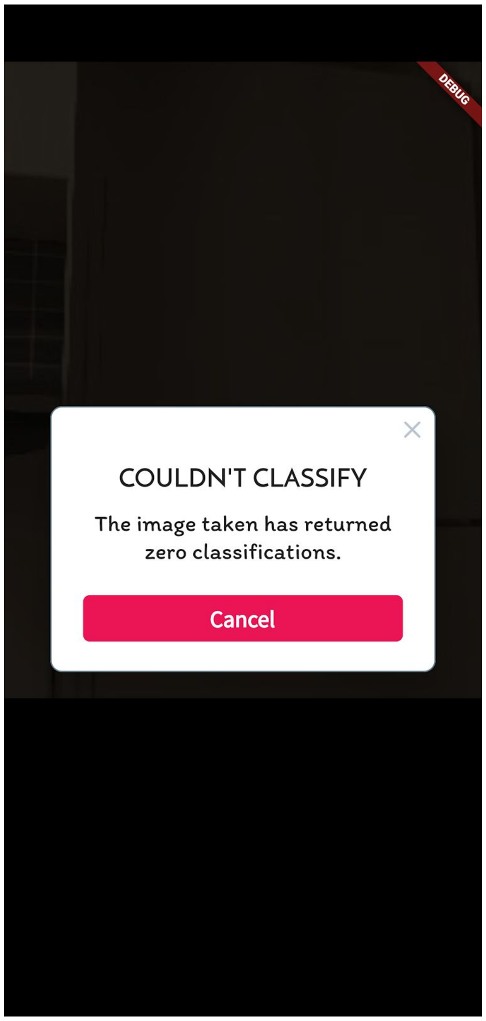
3. Captured Image Preview: This screen depicts the captured image and offers the user two options. If the image is not satisfactory or if it has not correctly captured the image, then an option is provided to *retake* the image and return to the previous image capture screen. Otherwise, the user can tap on the *search* button, and the app will redirect to the recommendations listing.



4. Generated Recommendations List: The recommendations generated from the captured image after it has been passed through the image processing unit (IPU) is shown in a list view for the user. The user can then select the classification that depicts the closest match. (In this example, we have selected '*tablet'* as the recommendation.)

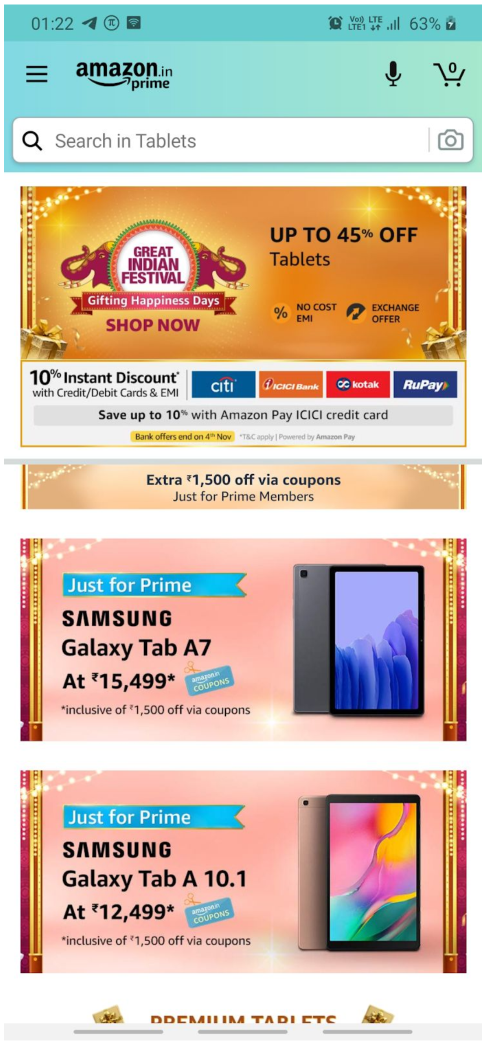


5. Unable to Classify: The implemented tensorflow image processing unit ensures that any classification that the model returns only those classifications that have a probability of at least 0.05. For this reason, there is a possibility that the model isn't able to identify any classification that has a sufficiently high probability. In this case, an alert is displayed that tells the user that the model wasn't able to classify. Upon pressing the Cancel button in the alert, the user is redirected to the home screen.

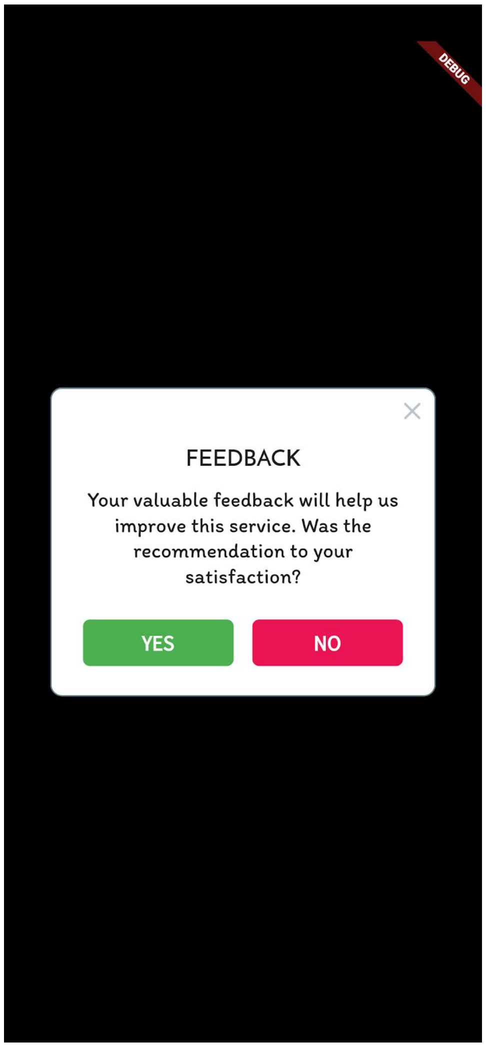

COULDN'T CLASSIFY

The image taken has returned zero classifications.

6. <u>Recommendation Display:</u> The text-based classification that the user selects is used to perform an API call to the google CustomSearch API with site-restrictions enabled so that only amazon.in is used for the search results. The URL thus obtained will then be used to open up a webview within the app itself. If for some reason, this is not possible, the URL is opened in the device's default browser.



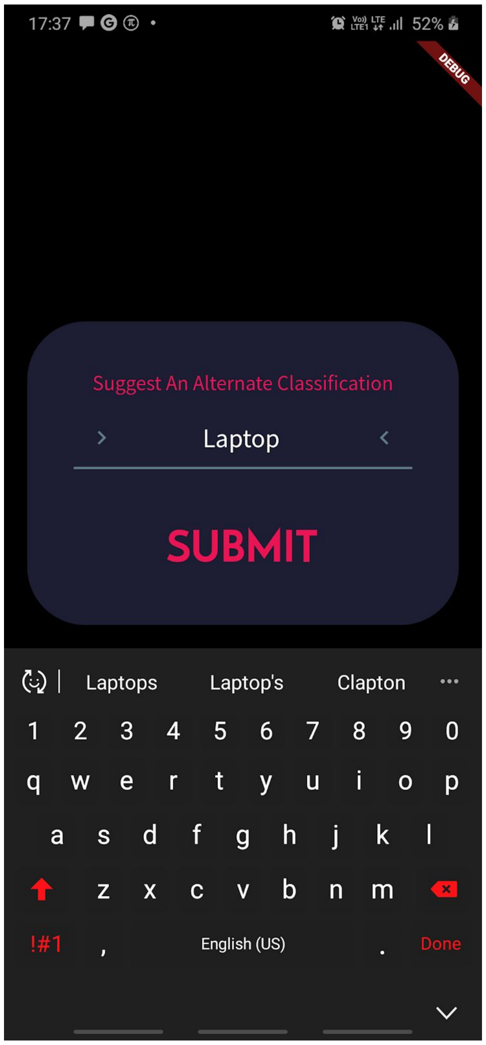
7. Asking for User's Feedback: When the user closes the webview and returns to the app, an alert immediately pops up asking the user for his feedback on whether the app has satisfied its purpose or if it can be improved. If the user is satisfied with the results, then he/she will be redirected to the home screen. If not, then the user is allowed to give his feedback.



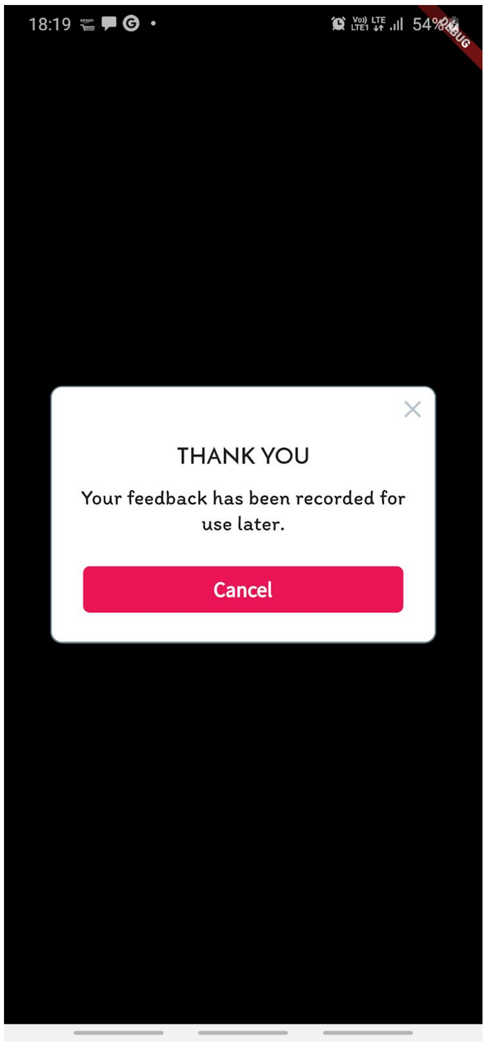
FEEDBACK

Your valuable feedback will help us improve this service. Was the recommendation to your satisfaction?

8. Feedback Screen: In this section of the app, the user is allowed to enter in a keyword / group of keywords that, along with the image and the app recommendation, gets stored in the app's firebase database, which will later on be used to modify / improve the user experience. The suggested recommendation string also goes through a cross checking mechanism to make sure that the suggested keyword is indeed valid.

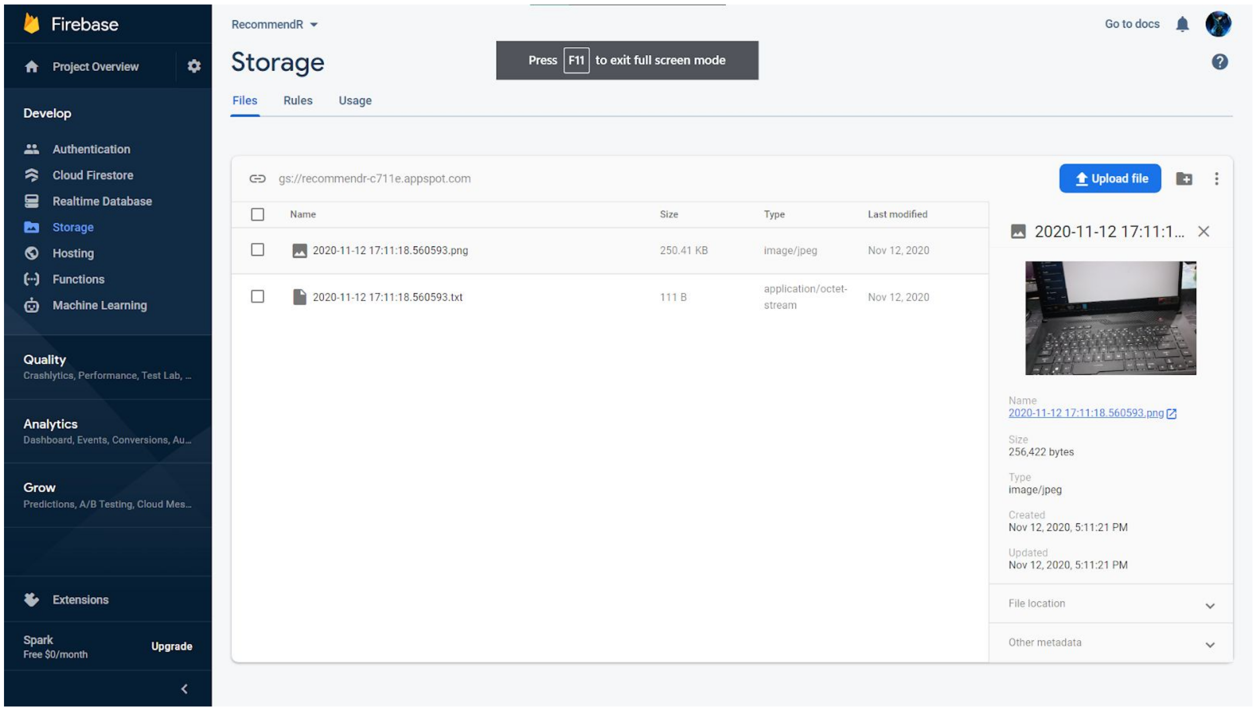


9. Thank You for the Feedback: Once the user submits the recommendation, the app redirects to a screen where the user is thanked for taking the time to provide the feedback necessary to maintain and improve our app and its user experience. The alert that pops up is automatically dismissed after a short duration, or the user can dismiss it simply by tapping on the *Cancel* button. Dismissing this screen redirects the user to the home screen.



Your feedback has been recorded for use later.

10. <u>User-Feedback gets stored in Firebase Storage:</u> Once the user provides feedback, the image as well as a text file (named using the current time) containing the app recommendation and user-suggested recommendation is uploaded to the Firebase Storage instance. This happens in the background and so, there is no delay in the working of the app.



Storage

gs://recommendr-c711e.appspot.com

Name	Size	Type	Last modified
2020-11-12 17:11:18.560593.png	250.41 KB	image/jpeg	Nov 12, 2020
2020-11-12 17:11:18.560593.txt	111 B	application/octet-stream	Nov 12, 2020

2020-11-12 17:11:1...

Name
2020-11-12 17:11:18.560593.png

Size
256,422 bytes

Type
image/jpeg

Created
Nov 12, 2020, 5:11:21 PM

Updated
Nov 12, 2020, 5:11:21 PM

Future Work:

There is scope for improvement in the image-based product recommender. For some images, it is not enough to classify it into its corresponding class, which may be too large at times. For example, if our app returns the classification *book*, that doesn't make any sense to an online shopper. It should return the title of the book if the recommendation is to be of any use. In such cases, an OCR text recognition algorithm can also be applied to improve our app. This functionality is not currently included in the design of the app as of now. After the specified requirements have been met, this functionality may be included if time permits.

The recommendation is shown to the user using Google's CustomSearch API. The free tier of this API allows for 100 API calls per day and anything more will require a monetary compensation. This is a student undertaking and as such does not have the funds nor the resources to properly utilize this API. Therefore, scaling the application is as of now, not feasible.

Another region where the app can be improved is the implementation of the feedback mechanism. Most users would not want to provide feedback and those that do will probably not want to type the recommendation out. So, providing a text field that would auto suggest products, from a product database, to the user will make the feedback mechanism much smoother. Also, a proper filter on the feedbacks that get stored onto the app's cloud maintained database is necessary to avoid the unnecessary wastage of resources.

To improve the image search we can use ideas mentioned in [2] and [5] where we maintain a database of feature vectors of product images on Amazon. In the case where the model predictions are incorrect or the input image does not belong to one of the supported product classes, we can employ this method. Here, we obtain the feature vector for the user query and rank the feature vectors in our database using a similarity metric. This is recommended to the user in such failure cases. In [4], they introduce the concept of model assertions to identify runtime errors such as this. We can also use Bayesian uncertainty estimates of the neural network output to determine when the model is not confident about its prediction. In such cases, we can trigger the implementations mentioned in [2] and [5].

References:

[1] S. Amershi et al., “Software Engineering for Machine Learning: A Case Study,” Proc. - 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract. ICSE-SEIP 2019, pp. 291–300, 2019.

[2] L. Chen, F. Yang, and H. Yang, “Image-based Product Recommendation System with CNN,” http://cs231n.stanford.edu/reports/2017/pdfs/105.pdf, p. 2015, 2015.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 4510–4520, 2019.

[4] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, “Model Assertions for Monitoring and Improving ML Models,” http://arxiv.org/abs/2003.01668, 2020.

[5] S. Bell and K. Bala, “Learning visual similarity for product design with convolutional neural networks,” ACM Trans. Graph., vol. 34, no. 4, 2015.

[6] T. Diethe, T. Borchert, E. Thereska, B. Balle, and N. Lawrence, “Continual Learning in Practice,” http://arxiv.org/abs/1903.05202, no. Nips, 2019.