# COMPUTER VISION BASED PRODUCT RECOMMENDER

**B.Tech CSE - V Sem - Software Engineering**
**CS300**

**VERSION 1 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.

## <u>MAIN TEXT</u>

### <u>Scope of the Work:</u>

This undertaking will involve the design of a mobile application that recommends products based on the image captured by the user. The application from the image data determines what product is to be searched for. Upon successful completion of the image processing, the app is redirected to a list of recommended products using Amazon APIs. If the recommendation was wrong or only partially correct, then, a mechanism is available to collect the feedback which is then used to refine the machine learning model in the backend.

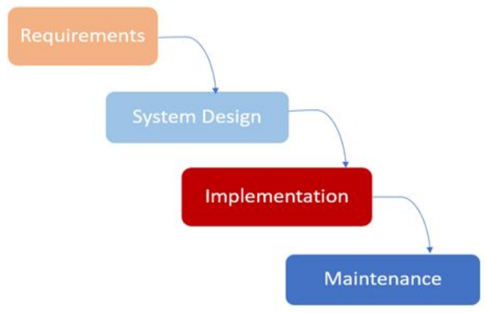
The application UI will be developed using flutter as it is a free and open source platform independent framework. The underlying image classification unit will use MobileNetV2 for processing the image and producing the recommendation. For some images, it is not enough to just classify it. For example, 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 mobile app that would perform the following tasks:

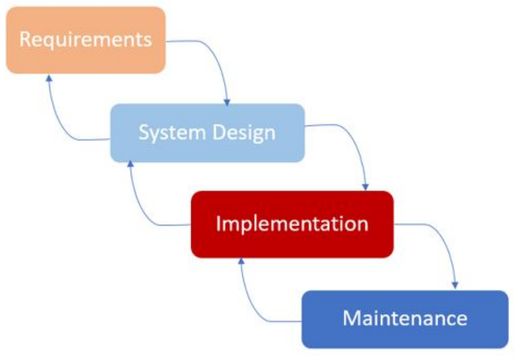
- ❏ Captures the image and feeds it to the image processing unit.
- ❏ The image classifier processes the image and outputs the classification of the captured image.
- ❏ Feedback mechanism which collects the feedback which is then used to refine the model.

**<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. In this paper, we explore how to incorporate existing software processes, related to machine learning and artificial intelligence [1], [5], [6], by modifying them to suit the development of our application.

**Working:**

The product does the following,

- ❑ It requires access to the camera of the mobile device used in order to get the image needed.
- ❑ It captures the image and feeds it to the Image Processing Unit (IPU) working in the backend.
- ❑ The IPU processes the image and outputs the text-based classification of the captured object.
- ❑ Then, feedback is collected and used to improve the model.

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

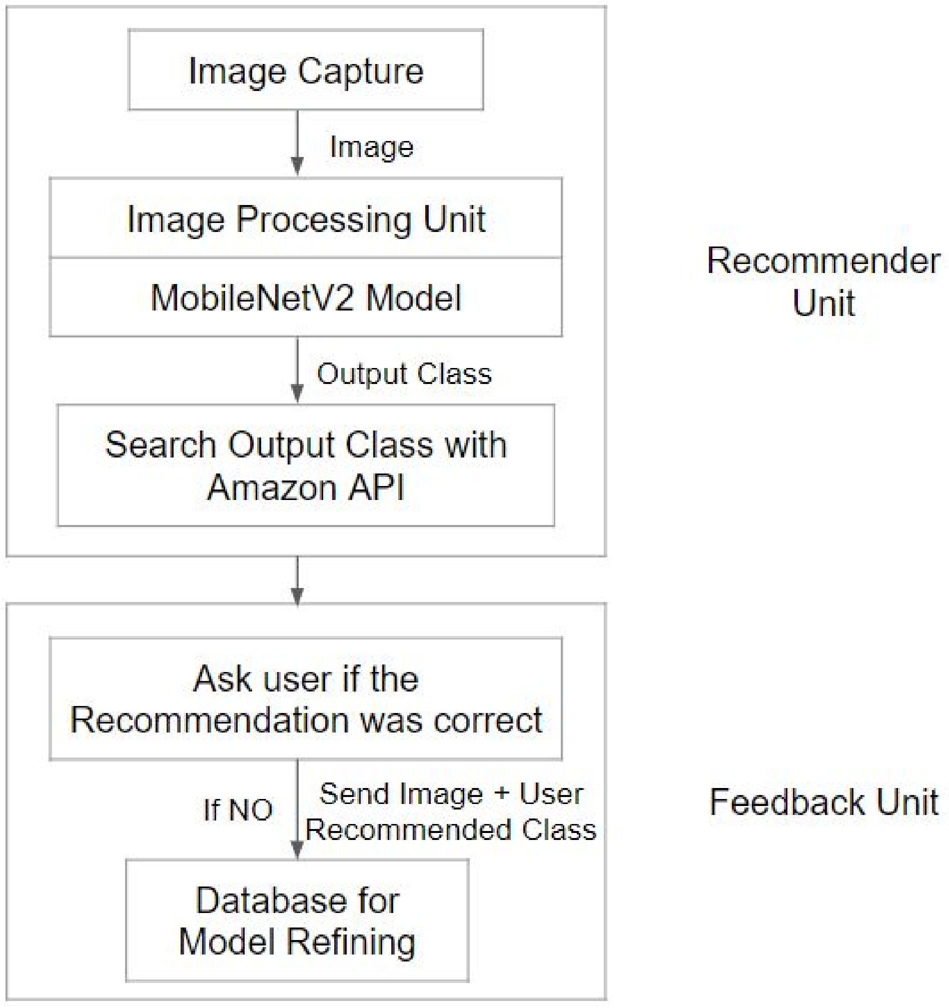


Image Capture
Image
Image Processing Unit
MobileNetV2 Model
Output Class
Search Output Class with Amazon API
Recommender Unit
Ask user if the Recommendation was correct
If NO
Send Image + User Recommended Class
Database for Model Refining
Feedback Unit

**<u>Platforms and Languages Used:</u>**

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

- ❏ Operating System : Android 8+ / IOS
- ❏ Frontend: Flutter/ Dart
- ❏ Backend: Firebase
- ❏ Image Processing Unit: Uses MobileNetV2 architecture [3] for the ML model

*<u>Why flutter?</u>*
Flutter is a free and open source Google mobile UI framework 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. Because it is platform independent, it can be used by both android and IOS developers with the same codebase.

*<u>Why MobileNetV2?</u>*
The IPU is run on device and hence networks that require high computational resources cannot be used. MobileNetV2 pushes the state of the art for mobile tailored computer vision models,by significantly decreasing the number of operations and memory needed while retaining the same accuracy, making it an apt choice for the machine learning architecture used.

**<u>Future Work:</u>**

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.

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].

**<u>References:</u>**

[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.