VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi, Karnataka-590018

A PROJECT REPORT ON

"MACHINE LEARNING BASED ANDROID GALLERY APPLICATION"

Submitted in partial fulfillment of the requirements of the Award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

MEGHANA U [4VV14CS048]

N S MEGHANA [4VV14CS050]

VINAY KRISHNA S [4VV14CS113]

VINAY V TRIMAL [4VV14CS114]

Under the Guidance of

POOJA M R

Associate Professor, Department of Computer Science & Engineering

VVCE, Mysuru

2017-2018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIDYAVARDHAKA COLLEGE OF ENGINEERING
P.B. No.206, Gokulam, III – Stage, MYSURU-570002

VIDYAVARDHAKA COLLEGE OF ENGINEERING
Gokulam 3rd Stage,
Mysuru-570002
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project entitled **"MACHINE LEARNING BASED ANDROID GALLERY APPLICATION"** is a bonafide work carried out by **MEGHANA U** (4VV14CS048), **N S MEGHANA** (4VV14CS050), **VINAY KRISHNA S** (4VV14CS113), **VINAY V TRIMAL (**4VV14CS114) of 8th semester Computer Science and Engineering in partial fulfillment for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University**, **Belgavi** during the academic year 2017-2018. It is certified that all the suggestions and corrections indicated for the internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the requirements in respect of project work prescribed for the said degree.

Signature of Guide	**Signature of HOD**	**Signature of Principal**
Pooja M R	**Dr Ravi Kumar V**	**Dr B Sadashive Gowda**
(Associate Professor,	(Professor & HOD,	(Principal,
Dept of CS&E,	Dept of CS&E,	VVCE,
VVCE, Mysuru)	VVCE, Mysuru)	Mysuru)

Name of the Examiners	**Signature with Date**
1)	
2)	

ACKNOWLEDGEMENT

If words are considered as the tokens of acknowledgements, then the words play the heralding role of expressing our gratitude.

With proud gratitude we thank God Almighty for all the blessings showered on us and for completing our project successfully.

We owe our gratitude to The Principal, **Dr. B Sadashive Gowda** for his whole hearted support and for his kind permission to undergo the project.

We wish to express our deepest gratitude to **Dr. Ravi Kumar V,** Head of Department, Computer Science and Engineering, VVCE, for his profound alacrity in our project and his valuable suggestions.

We wish to enunciate our special thanks to our paradigmatic and relevant Project coordinators **Natesh. M,** Associate Professor, **Karthik. V**, Assistant Professor, **Nandini. S. B**, Assistant Professor, and internal guide **Pooja. M. R,** Associate Professor, Department of Computer Science and Engineering, VVCE who gave us throughout our project period with their valuable suggestions and for devoting their precious time in making this project a success.

In the end, we offer our sincere thanks to our classmate and friend **Gururaj P**, for their valuable suggestions and all his help in implementing this project.

Meghana U (4VV14CS048)

N S Meghana (4VV14CS050)

Vinay Krishna S (4VV14CS113)

Vinay V Trimal (4VV14CS114)

ABSTRACT

Machine learning provides various techniques that have proven to be very effective in the domain of image processing. Android devices, specifically smartphones of recent times are equipped with a large internal storage space as a consequence of which, a bulk of data (such as images, video files, text files etc.) can be stored without having to worry about running out of storage space. Under these circumstances, sorting or organizing said image data based on their time of creation, access, or modification and according to the folders in which they are store in the device (as is done by conventional gallery applications) becomes cumbersome when one has to look for a particular image or a set of similar images. Thus, using machine learning techniques in a gallery application to classify and prioritize images would make the whole process more efficient as well as customized for each user, which otherwise would be impractical to explicitly program. From a commercial perspective this also serves as an incentive for users to use the application more, making it profitable to developers as well.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Conventional gallery applications in any typical android devices display and categorize the images based on time the image was created or the image was last accessed. These applications may also sort the images based on their locations in a file system. Users are allowed to perform few operations like edit, copy, delete or share multiple images at a time. Some other gallery applications might have an extra folder like recent to highlight the newly added images, but most of the gallery apps are simple and straightforward with the basic image folders. There can be too many folders, each of which containing multiple images. These image folders may take up the whole of available memory on the device which makes sorting, accessing or deleting the images inefficient at times. One possible approach to overcome this drawback is the use of machine learning techniques to organize the images. Machine learning uses statistical techniques to make systems the ability to learn with data without being explicitly programmed.

The use of machine learning in mobile applications brings in all those benefits that would consequently provide a better incentive to use the app on a daily basis. One such widely used application is the Google Photos app. It facilitates image backup from multiple devices in a single location, while also sorting pictures of the same people or objects or even the same location into organized groups. In addition to this, tasks like creating an album of taken during a specific period, such as on a vacation, are also automatically done. Photos also recognize faces that are present in the pictures frequently and sorts them accordingly. Other prominent features include Suggested Sharing and Shared Libraries. Suggested Sharing uses Google's machine learning algorithms to recommend photos that it thinks the user should share with certain people based on who is in the photos, where certain pictures or videos were captured, and who was with the user at that time. And with Shared Libraries, one can choose someone to share either their entire Google Photos collection with or just a specific album.

Other gallery applications that provide enhanced user experience in comparison with conventional apps are Focus -Picture Gallery and F-Stop Gallery. Focus features a tagging mechanism to access and organize albums. Each image can be associated with one or more 11 preset or additional user defined tags and can be accessed through them. F-Stop Gallery enables the user to instantly search through all their photos to find results based on file name and metadata (tags, ratings, camera model, etc.). It also allows the user to save tags (keywords) and ratings in XMP format so they stay with the image and can be read by popular programs such as Lightroom, Picasa, Aperture, Windows Live Photo Gallery, digiKam and many more. Further, it can read metadata directly from images (EXIF, XMP, and IPTC) and custom sort images using drag and drop. However, characteristic organization of images within these applications might be difficult.

In order to overcome the drawbacks of conventional gallery applications, we develop a customized gallery application by considering few parameters like number of times the image is viewed, average duration for which the image is viewed and average time between the consecutive accesses of images that would implicitly carry out the work of personalization by using machine learning to learn the uniqueness of each user and give the priority for the images which makes it easier to access/ delete if needed. These features make the application more efficient and user friendly. The use of machine learning would effectively improve the overall user experience and application usage.

CHAPTER 2

LITERATURE SURVEY

Sl. No	Author(s)	Paper Name	Description
1.	Siddhartha Sankar Nath, Jajnyaseni Kar, Girish Mishra, Sayan Chakraborty	A Survey of Image Classification Methods and Techniques	In this paper, we review of current activity of image classification methodologies and techniques. Image classification is a complex process which depends upon various factors. Here, we discuss about the current techniques, problems as well as prospects of image classification. The main focus will be on advanced classification techniques which are used for improving classification accuracy. Additionally, some important issues relating to classification performance.
2.	Yuguang Huang, Lei Li	Naive Bayes classification algorithm based on small sample set	Naive Bayes algorithm is one of the most effective methods in the field of text classification, but only in the large training sample set can it get a more accurate result. The requirement of a large number of samples not only brings heavy work for previous manual classification, but also puts forward a higher request for storage and computing resources during the computer post-processing. This paper mainly studies Naïve Bayes classification algorithm based on Poisson distribution model, and the experimental results show that this method keeps high classification accuracy even in small sample set.
3.	Qing-yun Dai, Chun-ping Zhang and Hao Wu.	Research of Decision Tree Classification Algorithm in Data Mining	Decision tree algorithm is one of the most important classification measures in data mining. Decision tree classifier as one type of classifier is a flow chart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node represents a class. The method that a decision tree model is used to classify a record is to find a path that from root to leaf by measuring the attributes test, and the attribute on the leaf is classification result.

4.	David Opitz and Richard Maclin.	Popular Ensemble Methods: An Empirical Study	An ensemble consists of a set of individually trained classifiers (such as neural networks or decision trees) whose predictions are combined when classifying novel instances. Previous research has shown that an ensemble is often more accurate than any of the single classifiers in the ensemble. Bagging (Breiman, 1996c) and Boosting (Freund & Schapire, 1996; Schapire, 1990) are two relatively new but popular methods for producing ensembles. Our work suggests that most of the gain in an ensemble's performance comes in the first few classifiers combined.
5.	Rui Xu and Donald Wunsch II	Survey of Clustering Algorithms	Data analysis plays an indispensable role for understanding various phenomena. Cluster analysis, primitive exploration with little or no prior knowledge, consists of research developed across a wide variety of communities. The diversity, on one hand, equips us with many tools. We survey clustering algorithms for data sets appearing in statistics, computer science, and machine learning, and illustrate their applications in some benchmark data sets, the traveling salesman problem, and bioinformatics, a new field attracting intensive efforts.
6.	Zhongjie Li and Rao Zhang	Object Detection and Its Implementation on Android Devices	Object detection is a very important task for different applications including autonomous driving, face detection, video surveillance, etc. CNN based algorithm could be a great solution for object detection with high accuracy. Considering there are a lot of mobile computing devices available, we implemented the CNN based object detection algorithm on Android devices. The model architecture is based on SqueezeNet to get image feature maps and a convolutional layer to find bounding boxes for recognized objects. The total model size is around 8 MB while most other object detection model takes more than 100 MB's storage. The model architecture makes the calculation more efficient, which enables its implementation on mobile devices.

7.	Amanpreet Singh and Narina Thakur and Aakanksha Sharma	A Review of Supervised Machine Learning Algorithms	Supervised machine learning is the construction of algorithms that are able to produce general patterns and hypotheses by using externally supplied instances to predict the fate of future instances. Supervised machine learning classification algorithms aim at categorizing data from prior information. Rule-based techniques, Logic-based techniques, Instance-based techniques, stochastic techniques. The main objective of this paper is to provide a general comparison with state of art machine learning algorithms.
8.	Durgesh K Srivastsa and Lekha Bhambhu	Data Classification using support vector machine	Classification is one of the most important tasks for different application such as text categorization, tone recognition, image classification, micro-array gene expression, proteins structure predictions, data Classification etc. SVM method does not suffer the limitations of data dimensionality and limited samples [1] & [2]. In this paper we have shown the comparative results using different kernel functions for all data samples.

CHAPTER 3

METHODOLOGY

This chapter deals with "methodology" which is popularly defined in two ways:

- “The analysis of the principles of methods, rules, and postulates employed by a discipline”.
- “The systematic study of methods that are, can be or have been applied within a discipline”.

The chapter describes how a conventional gallery app can be replaced by a machine learning based gallery application and how this can be achieved and also, why it should be done.

- The first section of the chapter describes how the conventional gallery application is rather inconvenient to the user and how machine learning is just the kind of thing that's needed to make this right.
- The second section describes the types of machine learning algorithms that are out there and these are the ones that need to be considered if a person needs to use it.
- The third section is about the algorithms that are suitable for our project based on certain intuitions which are mentioned in the section.
- The fourth section describes why we think naive Bayes classifier for probability estimation is the most suitable for our application.
- The fifth section gives a brief overview of that naive Bayes classifier algorithm and also describes how it can be used for probability estimation.

3.1 ADVANTAGES OF MACHINE LEARNING OVER AN EXPLICIT FUNCTION FOR PRIORITY

The objective is to limit the inconvenience of having to scroll through a ton of images before you actually find the image that you need. So, we want to make use of the usage statistics of the user and predict how important a particular image is to the corresponding user. Consider the scenario wherein there's a user and let's just say she doesn't possess a good memory. Every day before college he tends to look at the timetable image to know what classes will be there on that particular day. The timetable image on his phone is in the folder "WhatsApp images" and it arrived there months ago. Now, a conventional gallery application would sort the images in a way that the image most recently written into memory is displayed first (based on the write time). So the timetable image was written months ago and will, therefore, have the user scroll through tons of images which were written recently before he ends up at the timetable image. One would argue that he could shift the image to a different folder in which there are fewer images and manually make things more convenient. Let's just say he could. But that's just for one particular image. What if there are multiple images like that? Taking the trouble of manually organizing images is a tedious task.

The solution that we propose is to have a value associated with every image in a particular folder, called the priority value. The priority value signifies how important a particular image is to the user. The images displayed in a folder will, therefore, be displayed in a highest priority first order. This implies the user arrives at the intended image with minimal scrolling.
Now, the question is how to find the priority values for images in a folder. We could have an explicit function of the three features (count, duration, inner access time) like the sum of three values and treat this value as the priority. But this doesn't always work as the amount of influence every feature has, on the priority, depends on the person and therefore explicit functions with constant coefficients isn't a great option because it is more generic and doesn't take into consideration the uniqueness of the person.

This is where machine learning comes into play. We can essentially make the machine learn from the user (by making use of the user statistics such as count, duration, inter-access time) about how much influence a particular usage statistic or a feature has on the priority value. If we were to use a function with explicit values, then essentially that function would remain the same for all users Irrespective of their uniqueness. Using machine learning implies that the function that the machine has learned from the user respects her uniqueness.

3.2 TYPES OF MACHINE LEARNING ALGORITHMS

There some variations of how to define the types of Machine Learning Algorithms but commonly they can be divided into categories according to their purpose and the main categories are the following:

- Supervised learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

3.2.1 SUPERVISED LEARNING

The majority of practical machine learning uses supervised learning. Supervised learning [1] is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.
Supervised learning problems can be further grouped into regression and classification problems.

- Classification: A classification [3] problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems. [7]
- Random forest for classification and regression problems.
- Support vector machines for classification problems. [8]

3.2.2 UNSUPERVISED LEARNING

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there are no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Some popular examples of unsupervised learning algorithms are:

- k-means for clustering problems.
- A priori algorithm for association rule learning problems.

3.2.3 SEMI-SUPERVISED MACHINE LEARNING

Problems where you have a large amount of input data (X) and only some of the data is labelled (Y) are called semi-supervised learning problems.

These problems sit in between both supervised and unsupervised learning. A good example is a photo archive where only some of the images are labelled, (e.g. dog, cat, person) and the majority are unlabeled.

Many real-world machine learning problems fall into this area. This is because it can be expensive or time-consuming to label data as it may require access to domain experts. Whereas unlabeled data is cheap and easy to collect and store.

You can use unsupervised learning techniques to discover and learn the structure in the input variables. You can also use supervised learning techniques to make best guess predictions for the unlabeled data, feed that data back into the supervised learning algorithm as training data and use the model to make predictions on new unseen data.

3.2.4 REINFORCEMENT LEARNING

Reinforcement learning is a problem faced by an agent that learns behavior through trial and error interactions with the dynamic environment. The word reinforcement means different things in different fields like engineering, neuroscience, psychology, economics, and mathematics. Here, the reinforcement learning that we refer to has vague resemblance to work in psychology. The area is inspired by behavioral psychology, concerned with how agents ought to take actions so as to maximize some notion of cumulative reward. Here, the environment is typically formulated as a Markov decision process (MDP) as several dynamic programming techniques are made use of. Reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Instead, the focus is on online performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). Nowadays, Reinforcement learning (RL) is gaining popularity for its ability to solve complex sequential decision-making problems in control theory. It has attracted the attention of optimization theorists because of several popular success stories in the field of operations management. The central issues of reinforcement learning that are discussed are trading off exploration and exploitation, establishing the foundations to the field through Markov decision theory, learning from delayed reinforcement, constructing empirical models to accelerate learning, making use of generalization and hierarchy and coping with the hidden state. Reinforcement learning is a subset of machine learning which itself is a subset of Artificial Intelligence.

3.3 POTENTIALLY SUITABLE ALGORITHMS

In this section, we choose the algorithms [2] that can be considered for the purpose of our project in particular. We give a brief description of the input and the output that we intend and then, choose the model which best bridges this gap in the pipeline between the input and the output. In the machine learning jargon, that means we're describing the data (input) that is useful to train the model and the type of prediction that is intended from the model (output) and then choosing the best model which best serves the purpose.

We intend to collect data from the end user rather than using a dataset that's already out there. The data that we intend to collect are the usage statistics of the end user which includes

- The count - the number of times the user has explicitly opened the image.
- Average view duration - an average of the amounts of duration for which the user had viewed the particular image.
- Average inter-access time - Average of the differences in times of subsequent access. So, this is the data that is used to train the model.

On the other side, we would want the model to use these features to train itself and make predictions about the priority of images.

It is clear from the previous paragraph that this is a supervised learning problem. One of the first algorithms that we considered is the logistic regression, but we soon realized any regressor would not be suitable for this problem as for any given two values of two features, the value for the third feature is not unique. Therefore, we realized that this is surely not a regression problem and may potentially be a classification problem.

However, classification is basically dividing the number of data points into classes. This just doesn't suit our requirement perfectly. Even if we were to have a classifier, then we'll have to divide N number of data points into N number of classes and somehow have something tell us how one class would have more priority than the other. This isn't a sensible way of modelling because of the fact that as the ratio of the number of classes to the number of data points increases, the computational power needed also increases. Therefore, using a typical classification model would not help the cause. Which implies this is not a typical classification problem.

3.4 WHY IS NAIVE BAYES ALGORITHM SUITABLE?

At first, we considered regression and found no luck because the features were dependent on each other. Then we considered how using a typical classifier would mean we've to have as many classes as the number of images which is not desirable due to the fact that computational power required also increases and this would make it increasingly impossible to be carried out on-device. So, the problem here is that we want a continuous value as the prediction like in regression. What if we had the best of both worlds? What if there was some way for a classifier to predict a continuous output as the priority. That would be perfect for our application.

Naive Bayes algorithm [5] is just the thing that will fit in. Our features are independent of each other, which suits the naive Bayes algorithm perfectly. The "naive" in the "naive Bayes" refers to the fact that we assume the features are independent of each other in a rather naive sense. Also, naive Bayes is based on the Bayes theorem. Possibly, we can have all the features and train the model. Then, we can make the model predict the probability of how important the image is, given the values of features that were input.

The features (count, the average duration of view, average inter-access time) are actually independent of one another. How many times a person opens an image explicitly has absolutely nothing to do with how long she had viewed it on average. Likewise, how long a person views the image, is essentially independent of how frequently the subsequent accesses of the image that the user does. Therefore, we can conclude that the three features are independent of each other. This satisfies the main criteria for the Bayes algorithm. Hence, the probability of the intersection is just the multiplication of all the probabilities of individual features. This makes life a whole lot easier in a way that the computation of probabilities in that Bayes theorem equation becomes increasingly easy. This means we can train the naive Bayes model with all the data corresponding to that images that are opened considering them as the "important" class. And then use this model to obtain the priority value for each image, which is precisely the probability that a particular image, with a particular set of values for the features, belongs to the "important" class.

Bayes theorem finds many uses in the probability theory and statistics. This requires us to view things from a probabilistic point of view.

Bayes theorem lets us examine the probability of an event based on the prior knowledge of any event that related to the former event.

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}$$

Let A and B be two events

P(A|B) : the conditional probability that event A occurs , given that B has occurred. This is also known as the posterior probability.
P(A) and P(B) : probability of A and B without regard of each other.
P(B|A) : the conditional probability that event B occurs , given that A has occurred.

Take a simple machine learning problem, where we need to learn our model from a given set of attributes (in training examples) and then form a hypothesis or a relation to a response variable. Then we use this relation to predict a response, given attributes of a new instance. Using the Bayes theorem, its possible to build a learner that predicts the probability of the response variable belonging to some class, given a new set of attributes.

Based on the previous equation, we now assume that A is the response variable and B is the input attribute. We thus have

P(A|B) : conditional probability of response variable belonging to a particular value, given the input attributes. This is also known as the posterior probability.
P(A) : The prior probability of the response variable.
P(B) : The probability of training data or the evidence.
P(B|A) : This is known as the likelihood of the training data.

Therefore, the above equation can be rewritten as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

The naive Bayes algorithm reduces the complexity of a classifier by making an assumption of conditional independence over the training dataset.

The assumption of conditional independence states that, given random variables X, Y and Z, we say X is conditionally independent of Y given Z, if and only if the probability distribution governing X is independent of the value of Y given Z.

This assumption makes the Bayes algorithm, naive.

Given, n different attribute values, the likelihood now can be written as

$$P(X_1 \ldots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

Here, X represents the attributes or features, and Y is the response variable. Now, P(X|Y) becomes equal to the products of, probability distribution of each attribute X given Y.

Finding the posterior probability or P(Y|X) is of interest here. Now, for multiple values of Y, we will need to calculate this expression for each of them.

Given a new instance Xnew, we need to calculate the probability that Y will take on any given value, given the observed attribute values of Xnew and given the distributions P(Y) and P(X|Y) estimated from the training data.

We will thus predict the class of the response variable, based on the different values we attain for P(Y|X). We simply take the most probable or maximum of these values. Therefore, this procedure is also known as maximizing a posteriori (maximum posteriori in bold).
Maximizing Likelihood

If we assume that the response variable is uniformly distributed, that is it is equally likely to get any response, then we can further simplify the algorithm. With this assumption the priori or P(Y) becomes a constant value, which is 1/categories of the response.As, the priori and evidence are now independent of the response variable, these can be removed from the equation. Therefore, the maximizing the posteriori is reduced to maximizing the likelihood problem.

As seen above, we need to estimate the distribution of the response variable from training set or assume uniform distribution. Similarly, to estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set. Such assumptions are known as event models. The variations in these assumptions generates different algorithms for different purposes. For continuous distributions, the Gaussian naive Bayes is the algorithm of choice. For discrete features, multinomial and Bernoulli distributions as popular.

Naive Bayes classifiers work really well in complex situations, despite the simplified assumptions and naivety. The advantage of these classifiers is that they require small number of training data for estimating the parameters necessary for classification. This is the algorithm of choice for text categorization. This is the basic idea behind naive Bayes classifiers

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 HARDWARE SPECIFICATION (for development):

- Intel Core2duo or above
- 4GB RAM (min), 12GB & above (ideal)
- 50GB Hard disk or above
- Standard Keyboard, Mouse, Monitor

4.2 SOFTWARE SPECIFICATION (for development):

- Operating System: Windows 7 & above or any Linux distribution
- Front end: Android Studio
- Back end: SQLite
- Coding language: Android, Java

4.3 DEPLOYMENT

- Any Android phone

CHAPTER 5

SYSTEM DESIGN

The proposed gallery application would use/employ ML to prioritize images in a folder according to the user's changing preferences by using information about how often and how long the an image is viewed. This information is then used to derive the feature vector that will be used to train the ML model. As mentioned before, this technique is much more efficient than typical sorting methods based on image names or access times.

Activity: An activity provides the window in which the app draws its UI. This window typically fills the screen, but may be smaller than the screen and float on top of other windows. Generally, one activity implements one screen in an app. For instance, one of an app's activities may implement a *Preferences* screen, while another activity implements a *Select Photo* screen. Thus, every time an image is opened it is considered to be a new activity and the closing of an image would indicate the end of that activity consequently causing a change in activity.

Over the course of its lifetime (Fig 5.1), an activity goes through a number of states. You use a series of callbacks to handle transitions between states as shown in the figure. We define the functions onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), and onDestroy() as required for the application.

onCreate():

This callback fires when the system first creates the activity. On activity creation, the activity enters the Created state. In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity. This method receives the parameter savedInstanceState, which is a Bundle object containing the activity's previously saved state. If the activity has never existed before, the value of the Bundle object is null.

onStart():

When the activity enters the Started state, the system invokes this callback. The onStart() call makes the activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive. For example, this method is where the app initializes the code that maintains the UI.

When the activity moves to the started state, any lifecycle-aware component tied to the activity's lifecycle will receive the ON_START event.

The onStart() method completes very quickly and, as with the Created state, the activity does not stay resident in the Started state. Once this callback finishes, the activity enters the Resumed state, and the system invokes the onResume() method.

onResume():

When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user. The app stays in this state until something happens to take focus away from the app. Such an event might be, for instance, receiving a phone call, the userâ€™s navigating to another activity, or the device screenâ€™s turning off.

When the activity moves to the resumed state, any lifecycle-aware component tied to the activity's lifecycle will receive the ON_RESUME event. This is where the lifecycle components can enable any functionality that needs to run while the component is visible and in the foreground, such as starting a camera preview.

When an interruptive event occurs, the activity enters the Paused state, and the system invokes the onPause() callback.

If the activity returns to the Resumed state from the Paused state, the system once again calls onResume() method. For this reason, you should implement onResume() to initialize components that you release during onPause(), and perform any other initializations that must occur each time the activity enters the Resumed state.

onPause():

The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed); it indicates that the activity is no longer in the foreground (though it may still be visible if the user is in multi-window mode). Use the onPause() method to pause or adjust operations that should not continue (or should continue in moderation) while the Activity is in the Paused state, and that you expect to resume shortly. There are several reasons why an activity may enter this state.

When the activity moves to the paused state, any lifecycle-aware component tied to the activity's lifecycle will receive the ON_PAUSE event. This is where the lifecycle components can stop any functionality that does not need to run while the component is not in the foreground, such as stopping a camera preview.

You can also use the onPause() method to release system resources, handles to sensors (like GPS), or any resources that may affect battery life while your activity is paused and the user does not need them. However, as mentioned above in the onResume() section, a Paused activity may still be fully visible if in multi-window mode. As such, you should consider using onStop() instead of onPause() to fully release or adjust UI-related resources and operations to better support multi-window mode.

onStop():

When your activity is no longer visible to the user, it has entered the Stopped state, and the system invokes the onStop() callback. This may occur, for example, when a newly launched activity covers the entire screen. The system may also call onStop() when the activity has finished running, and is about to be terminated.

When the activity moves to the stopped state, any lifecycle-aware component tied to the activity's lifecycle will receive the ON_STOP event. This is where the lifecycle components can stop any functionality that does not need to run while the component is not visible on the screen.

In the onStop() method, the app should release or adjust resources that are not needed while the app is not visible to the user. For example, your app might pause animations or switch from fine-grained to coarse-grained location updates. Using onStop() instead of onPause() ensures that UI-related work continues, even when the user is viewing your activity in multi-window mode.

onDestroy():

onDestroy() is called before the activity is destroyed. The system invokes this callback either because:

* The activity is finishing (due to the user completely dismissing the activity or due to finish() being called on the activity), or
* The system is temporarily destroying the activity due to a configuration change (such as device rotation or multi-window mode)

When the activity moves to the stopped state, any lifecycle-aware component tied to the activity's lifecycle will receive the ON_DESTROY event. This is where the lifecycle components can clean up anything it needs to before the Activity is destroyed. The onDestroy() callback should release all resources that have not yet been released by earlier callbacks such as onStop().

In the proposed system, we associate the application with a database containing the attributes shown Table 5.1 to keep track of the usage statistics of images on the device. Among these, the count, avgDuration, and avgInterAccess are used to derive features to train the ML model. Prior to the initial run of the application, the images would be displayed in an *nx2* grid (where *n*, the number of rows, depends on the number of images on the device) in a default order determined by the image names. This database is continually updated every time the application is run. The setting up of this database thus serves as the first stage of system design.

Table 5.1: Empty database

Attribute	**Img_Name**	**Img_Path**	**Count**	**Avg Duration**	**Last Access**	**Avg Inter-access**	**Priority**
Initial Value	-	-	1	0.0	0.0	0.0	0.0

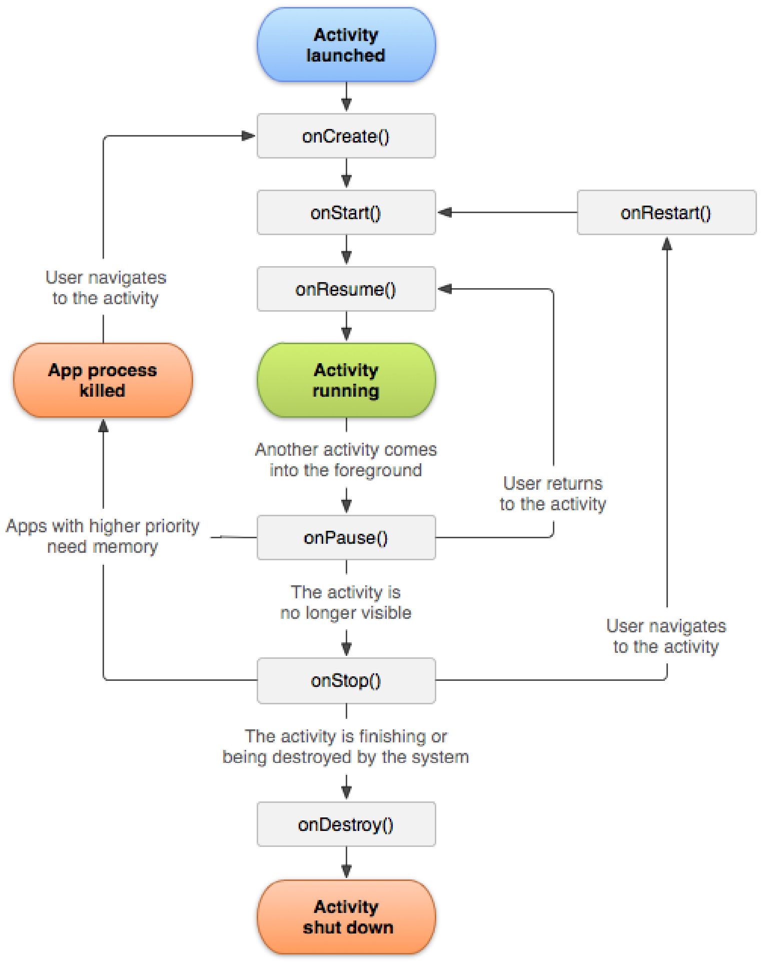

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Fig 5.1: Android Activity Life Cycle

The second step in the design is to update and record the attribute values of the images in the database every time an image is opened. This is done in the following order:

- The count of how many times the image is viewed is the first attribute to be updated.

- To record each instance of the duration for which an image is viewed, a timer, which starts when the image is opened and ends when the image changes/is closed, is used. This value is then used to update the average duration

- Finally, in order to obtain the average inter access time of an image, the last access time of the image is stored every time it is opened and the average is calculated using the current access time and the last access time of that particular image.

These values are updated as long as the application runs in the foreground. When it is closed, this data is used to generate a feature vector which is then stored in a text file. This file is then passed as the dataset to the Naive Bayes algorithm which processes it and generates an array of priorities as the output.

This array is then used to update the priority attribute of each image in the database following which, a query is run to display the images in the decreasing order of their priority the next time the application is run. This entire process is shown in Fig 5.2.

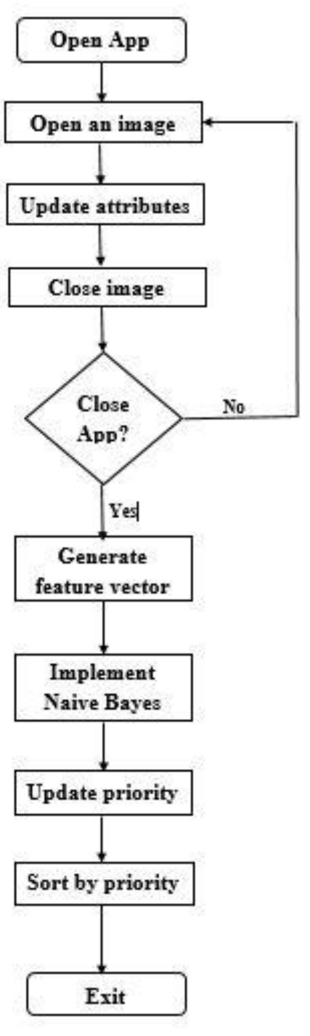


Fig 5.2: System Design

CHAPTER 6

IMPLEMENTATION

The proposed gallery application is implemented by first setting up a database associated with the application, to record and store the required usage statistics of each image in the database. Among these, the count, avgDuration, and avgInterAccess are used to derive features to train the ML model. This database is continually updated every time the application is run.

Table 6.1: Image database

Attribute	**Img_Name**	**Img_Path**	**Count**	**Avg Duration**	**Last Access**	**Avg Inter-access**	**Priority**
Initial Value	-	-	1	0.0	0.0	0.0	0.0

Every time an image is opened, its attribute values are updated in the following order:

- The count is the first attribute to be updated in the onPause() callback. It is done as follows:

  count = count + 1

- As mentioned before, the opening of an image is considered a new activity and when this activity begins, a timer is started. This timer is defined in the onPause() and onResume() callbacks of the activity. The time at which this timer begins is also noted (as the CurrentAccess). Using the duration of the timer obtained in each instance, the average duration of view of each image in the database is calculated as follows: avgDuration = ((avgDuration x (count - 1)) + timerValue) / count

- When an image is closed, marking the end of that activity, its time of CurrentAccess is stored in the database as its LastAccess. Thus, using the updated CurrentAccess when the image is opened, and its LastAccess time retrieved from the database, we calculate the average inter access time for each image in the database as follows: avgInterAccess = ((avgInterAccess x (count - 2)) + (CurrentAccess - LastAccess)) / (count - 1)

While these values are updated continually (Fig 7.1 and Fig 7.3) when the application is running in the foreground, when it is closed, this information is used to generate the feature vector to be passed as input to the Naïve Bayes algorithm. For this, the attributes count, avgDuration, and avgInterAccess are considered. The maximum and minimum values in the range of each of these attributes are used to divide the range into three equal intervals. The upper and lower limits of these intervals are used to determine offset values that decide the binary value in the feature vector for that image. The three features considered are thus expanded to obtain nine values in the vector. This is done as follows:

countOffset1 = minCount + (maxCount - minCount) / 3; 1.1
countOffset2 = minCount + (2 x (maxCount - minCount) / 3); 1.2
durationOffset1 = minTime + (maxDuration - minDuration) / 3; 1.3
durationOffset2 = minTime + (2 x (maxDuration - minDuration) / 3); 1.4
interaccessOffset1 = minInter + (maxInter - minInter) / 3; 1.5
interaccessOffset1 = minInter + (2 x (maxInter - minInter) / 3); 1.6

The values of the attributes in the database are compared with these offsets to obtain binary values which are then written into the text file to be passed as input to the ML model. This text file (Fig 7.2 and Fig 7.4) will contain the following data:

- The first line of the file will contain the number of features in the feature vector. In this case, it is 9.
- The second line of the file will contain the number of entries i.e. the number of images in the folder.
- The consequent lines each will contain 9 binary values followed by a colon ':' and then he label '1'. Out of the 9 binary values, the first 3 are determined by the count, the net 3 by the average duration of viewing, and the last 3 by the average inter-access time.
    - If the count/average duration of the image is greater than all 3 offset values, then the value will be '1 1 1', if it is greater than only the first two offsets, the value will be '1 1 0', and if it is greater than only the minimum offset, the value will be '1 0 0'.

- If the average inter access time of the image less than all 3 offset values, then the value will be ‘1 1 1’, if it is less than only the first two offsets, the value will be ‘1 1 0’, and if it is less than only the minimum offset, the value will be ‘1 0 0’.

As mentioned already, the naive Bayes algorithm has been chosen to predict the priority. We converted the values of the features to binary values because naive Bayes needs independent conditional probabilities rather than a continuous set of values. This is the reason we divided the whole range of the features into three equal parts and converted the 3 feature values into 9 binary "yes or no" conditions as already mentioned in the former part of this chapter.

We consider these as 9 values for conditional probabilities signified in a generic notation as P(feature/importance) and these binary values are input into the naive Bayes code (Fig 7.5). After the pre-processing of data, it is used to train the naive Bayes model which is then used to predict a priority value (Probability of importance) for a particular set of feature values of an image.

Under the hood, once the values are stored in the database as mentioned in the former part of the chapter, this is used to write to a text file using the append() method of the class OutputStreamWriter. This text file which contains the training for is then input into the model wherein it is read using the parseInt method of the class Integer (one particular value after the other).The trainNaiveBayes() function in the code of the naive Bayes model increments the posCount whenever it encounters a data point whose label is 1 (which is basically every data point as we're modeling only label 1). It also find the number of 1's present for each feature in all data points and stores these values in an array called featureCountPos[ ].

The priority is predicted by the getClassification () (Fig 7.7) function. For every subsequent value of the array, if the value is 1 then the corresponding value of featureCountPos[] array , or if the value of featureVector[] is 0, then the posClass is assigned with the difference of posCount and the corresponding value in the featureCountPos[] array which is the sum of all 1’s seen by the model. The value of log of the value obtained by dividing posClass by the posCount is added to the variable logProbPos in every iteration. The value of logProbPos at the end of all the iterations of the featureVector[] is output as that priority of the image.

CHAPTER 7

RESULTS & SNAPSHOTS

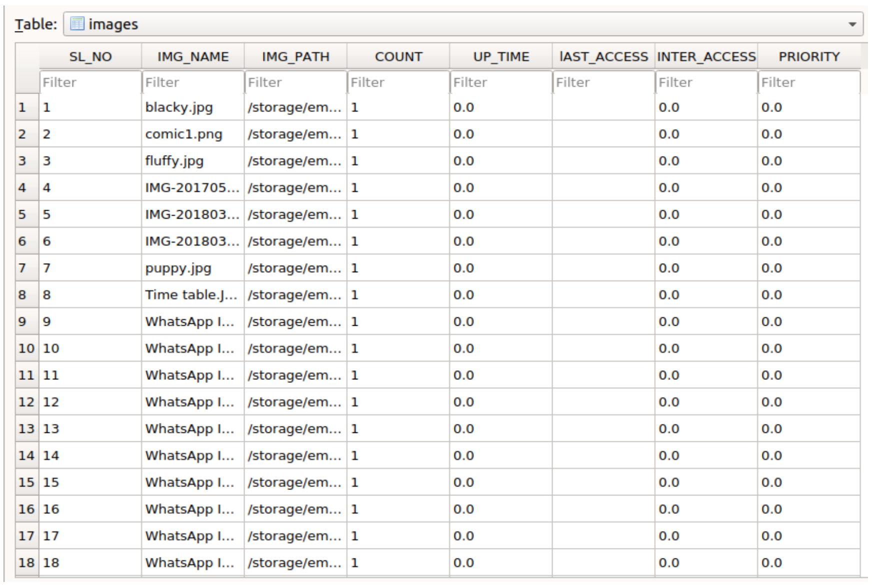
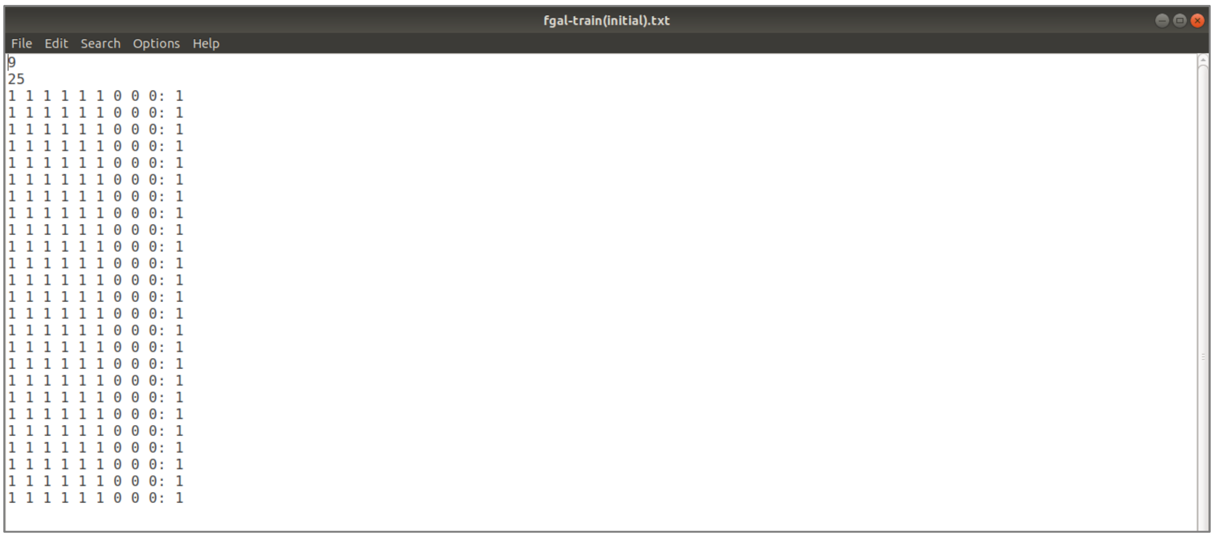


Table: images

	SL_NO	IMG_NAME	IMG_PATH	COUNT	UP_TIME	lAST_ACCESS	INTER_ACCESS	PRIORITY
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	blacky.jpg	/storage/em...	1	0.0		0.0	0.0
2	2	comic1.png	/storage/em...	1	0.0		0.0	0.0
3	3	fluffy.jpg	/storage/em...	1	0.0		0.0	0.0
4	4	IMG-201705...	/storage/em...	1	0.0		0.0	0.0
5	5	IMG-201803...	/storage/em...	1	0.0		0.0	0.0
6	6	IMG-201803...	/storage/em...	1	0.0		0.0	0.0
7	7	puppy.jpg	/storage/em...	1	0.0		0.0	0.0
8	8	Time table.J...	/storage/em...	1	0.0		0.0	0.0
9	9	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
10	10	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
11	11	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
12	12	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
13	13	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
14	14	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
15	15	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
16	16	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
17	17	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0
18	18	WhatsApp I...	/storage/em...	1	0.0		0.0	0.0

Fig 7.1: Database before any image is opened



```
9
25
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 1 0 0 0: 1
```

Fig 7.2: Text file before any image is opened

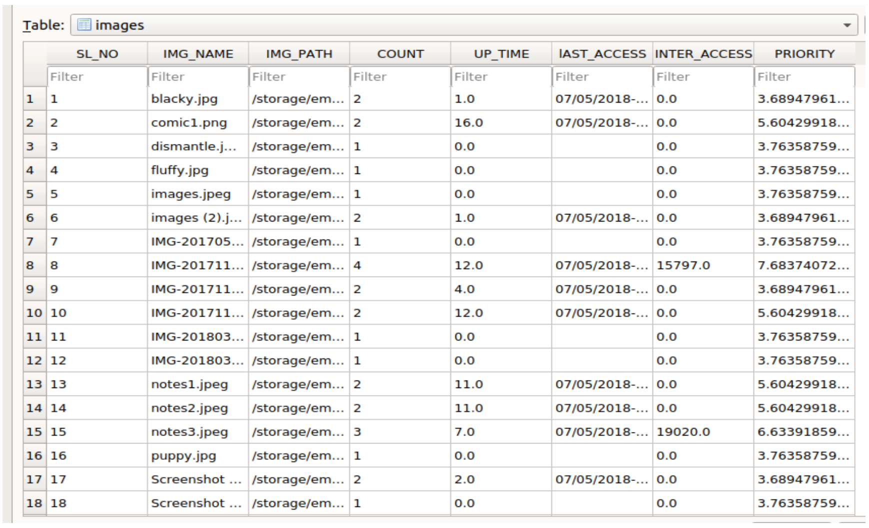
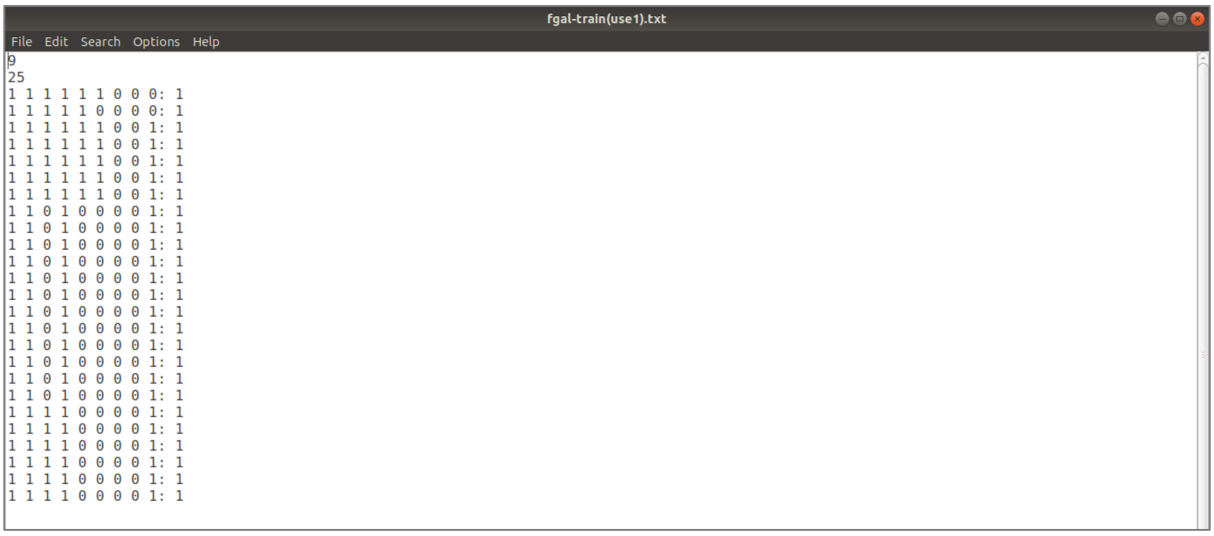


Table: images

	SL_NO	IMG_NAME	IMG_PATH	COUNT	UP_TIME	lAST_ACCESS	INTER_ACCESS	PRIORITY
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	blacky.jpg	/storage/em...	2	1.0	07/05/2018-...	0.0	3.68947961...
2	2	comic1.png	/storage/em...	2	16.0	07/05/2018-...	0.0	5.60429918...
3	3	dismantle.j...	/storage/em...	1	0.0		0.0	3.76358759...
4	4	fluffy.jpg	/storage/em...	1	0.0		0.0	3.76358759...
5	5	images.jpeg	/storage/em...	1	0.0		0.0	3.76358759...
6	6	images (2).j...	/storage/em...	2	1.0	07/05/2018-...	0.0	3.68947961...
7	7	IMG-201705...	/storage/em...	1	0.0		0.0	3.76358759...
8	8	IMG-201711...	/storage/em...	4	12.0	07/05/2018-...	15797.0	7.68374072...
9	9	IMG-201711...	/storage/em...	2	4.0	07/05/2018-...	0.0	3.68947961...
10	10	IMG-201711...	/storage/em...	2	12.0	07/05/2018-...	0.0	5.60429918...
11	11	IMG-201803...	/storage/em...	1	0.0		0.0	3.76358759...
12	12	IMG-201803...	/storage/em...	1	0.0		0.0	3.76358759...
13	13	notes1.jpeg	/storage/em...	2	11.0	07/05/2018-...	0.0	5.60429918...
14	14	notes2.jpeg	/storage/em...	2	11.0	07/05/2018-...	0.0	5.60429918...
15	15	notes3.jpeg	/storage/em...	3	7.0	07/05/2018-...	19020.0	6.63391859...
16	16	puppy.jpg	/storage/em...	1	0.0		0.0	3.76358759...
17	17	Screenshot ...	/storage/em...	2	2.0	07/05/2018-...	0.0	3.68947961...
18	18	Screenshot ...	/storage/em...	1	0.0		0.0	3.76358759...

Fig 7.3: Updated database



```
9
25
1 1 1 1 1 1 0 0 0: 1
1 1 1 1 1 0 0 0 0: 1
1 1 1 1 1 1 0 0 1: 1
1 1 1 1 1 1 0 0 1: 1
1 1 1 1 1 1 0 0 1: 1
1 1 1 1 1 1 0 0 1: 1
1 1 1 1 1 1 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 0 1 0 0 0 0 1: 1
1 1 1 1 0 0 0 0 1: 1
1 1 1 1 0 0 0 0 1: 1
1 1 1 1 0 0 0 0 1: 1
1 1 1 1 0 0 0 0 1: 1
1 1 1 1 0 0 0 0 1: 1
1 1 1 1 0 0 0 0 1: 1
```

Fig 7.4: Updated text file

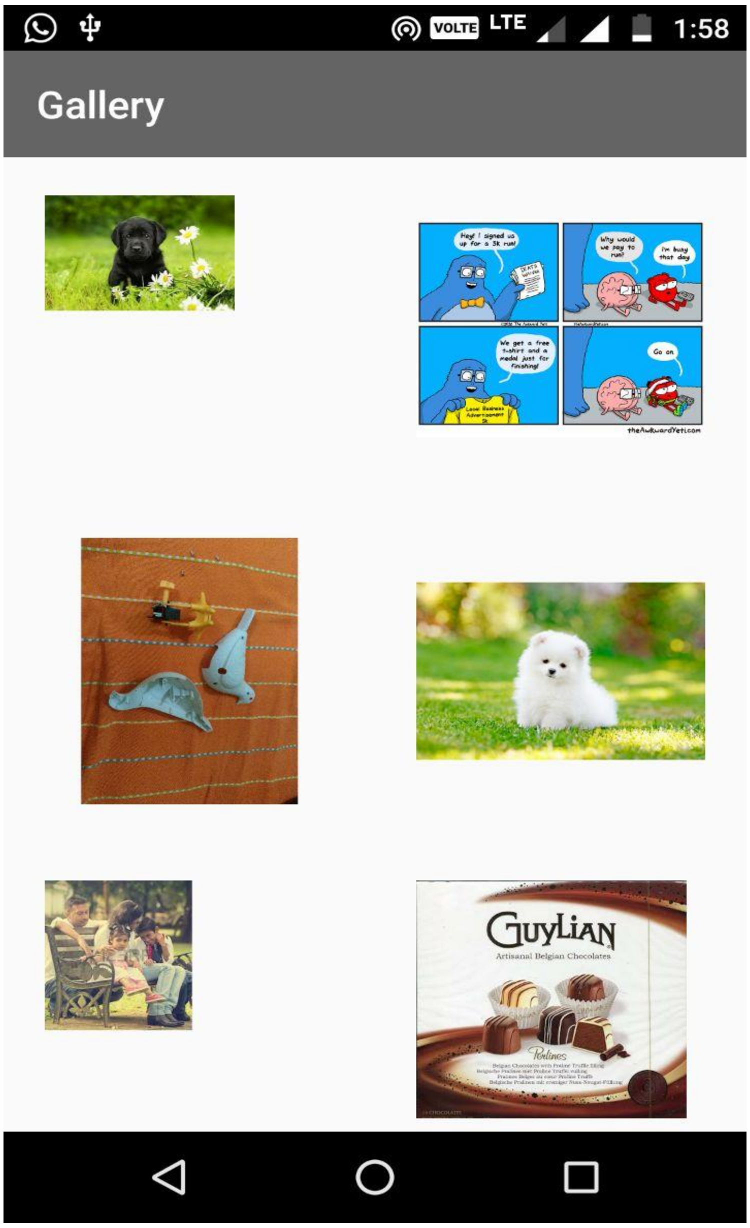


Fig 7.5: Gallery before prioritization

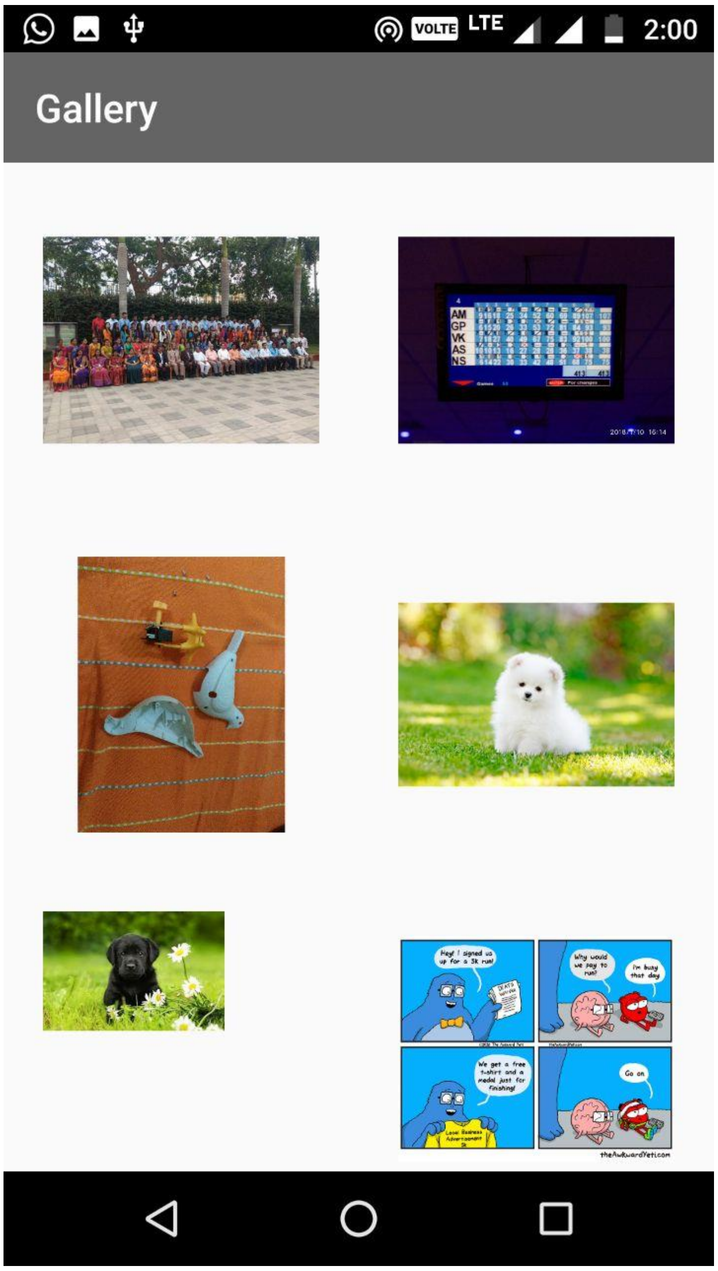


Fig 7.6: Gallery after prioritization

CONCLUSION

Through the proposed system, we intend to develop a customized gallery application. This application will prioritize the images based on the features which make it easier to access/ delete if needed. The features make the application more efficient and user friendly compared to other typical gallery applications. Also, the application will implicitly carry out the work of personalization using machine learning which will greatly affect the overall user experience.

FUTURE ENHANCEMENT

One of the simplest ways to make the application more efficient is to increase the number of features taken into account. This could include data about the type of image, quality, image source, color composition, presence/absence of editing filters, text in the image, etc. When the number of features increase, the standard of the feature vector too increases, thus making priority prediction more accurate.

Further, along with prioritizing the images, if they can also be classified into a number of classes, it would make it easier for the user to access similar images from different folders together. Such classification would also make it easier to filter out redundant images on the device. For the purpose of classification, there already exist numerous trained models which can be deployed as per requirement. For example, if a user has pictures of various food stored across different folders, classifying and gathering all food related images first and then prioritizing them would make it easier to get rid of unused or redundant images from the device.

REFERENCES

[1] Amanpreet Singh, Narina Thakur & Aakanksha Sharma, “A Review of Supervised Machine Learning Algorithms”, IEEE-2016, International Conference on Computing for Sustainable Global Development (INDIACom).

[2] David Opitz & Richard Maclin, “Popular Ensemble Methods: An Empirical Study”, Journal of Artificial Intelligence Research 11 (1999) 169-198.

[3] Siddhartha Sankar Nath, Jajnyaseni Kar, Girish Mishra, Sayan Chakraborty & Nilanjan Dey, “A Survey of Image Classification Methods and Techniques”, 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).

[4] Zhongjie Li, Rao Zhang, “Object Detection and Its Implementation on Android Devices”.

[5] Yuguang Huang & Lei Li, “Naive Bayes Classification Algorithm Based on Small Sample Set”, Proceedings of IEEE CCIS-2011.

[6] Rui Xu, “Survey of Clustering Algorithms”, IEEE Transactions on Neural Networks, Vol. 16, No. 3, May 2005.

[7] Qing-yun Dai, Chun-ping Zhang & Hao Wu, “Research of Decision Tree Classification Algorithm in Data Mining”, International Journal of Database Theory and Application, Vol.9, No.5 (2016), pp.1-8.

[8] Durgesh K. Srivastava & Lekha Bhambhu, “Data Classification using Support Vector Machine”, Journal of Theoretical and Applied Information Technology, 2005 - 2009 JATIT.

Machine Learning Based Android Gallery Application

N.S. Meghana
Department of Computer Science and Engineering
Vidyavardhaka College of Engineering
Mysuru, Karnataka, India
E-mail: nsmeghana01@gmail.com

Meghana. U
Department of Computer Science and Engineering
Vidyavardhaka College of Engineering
Mysuru, Karnataka, India
E-mail: meghanachethana@gmail.com

Vinay. V. Trimal
Department of Computer Science and Engineering
Vidyavardhaka College of Engineering
Mysuru, Karnataka, India
E-mail: vinaytrimal96@gmail.com

Vinay Krishna. S
Department of Computer Science and Engineering
Vidyavardhaka College of Engineering
Mysuru, Karnataka, India
E-mail: vinaykrishna21@gmail.com

Pooja. M. R
Associate Professor
Department of Computer Science and Engineering
Vidyavardhaka College of Engineering
Mysuru, Karnataka, India
Email: pooja.mr@vvce.ac.in

Abstract: Machine learning provides various techniques that have proven to be very effective in the domain of image processing. Android devices, specifically smartphones of recent times are equipped with a large internal storage space as a consequence of which, a bulk of data (such as images, video files, text files etc.) can be stored without having to worry about running out of storage space. Under these circumstances, sorting or organizing said image data based on their time of creation, access, or modification and according to the folders in which they are store in the device (as is done by conventional gallery applications) becomes cumbersome when one has to look for a particular image or a set of similar images. Thus, using machine learning techniques in a gallery application to classify and prioritize images would make the whole process more efficient as well as customized for each user, which otherwise would be impractical to explicitly program. From a commercial perspective this also serves as an incentive for users to use the application more, making it profitable to developers as well.

Keywords: Machine Learning, Classification, Prioritization, Inception, Regression.

I. INTRODUCTION

Conventional gallery applications in android devices display and sort images based on time (created/ last accessed). Users are allowed to share, copy, delete, or edit the images. These image folders may contain a large number of images which makes accessing/ deleting the images inefficient at times. One possible approach to overcome this drawback is the use of machine learning techniques to organize the images. This would greatly improve the overall user experience and application usage.

II. RELATED WORK

Machine learning was mainly employed in recommender systems used by businesses (especially e-commerce industries) with the aim of enhancing user experience through personalization. Using machine learning in mobile applications brings in all those benefits that would consequently provide a better incentive to use the app on a daily basis.

One such widely application is the Google Photos app. It facilitates image backup from multiple devices in a single location, while also sorting pictures of the same people or objects or even the same location into organized groups. In addition to this, tasks like creating an album of taken during a specific period, such as on a vacation, are also automatically done. Photos also recognize faces that are present in the pictures frequently and sorts them accordingly. Other prominent features include Suggested Sharing and Shared Libraries. Suggested Sharing uses Google's machine learning algorithms to recommend photos that it thinks the user should share with certain people based on who is in the photos, where certain pictures or videos were captured, and who was with the user at that time. And with Shared Libraries, one can choose someone to share either their entire Google Photos collection with or just a specific album.

Other gallery applications that provide enhanced user experience in comparison with conventional apps are Focus - Picture Gallery and F-Stop Gallery. Focus features a tagging mechanism to access and organize albums. Each image can be associated with one or more 11 preset or additional user defined tags and can be accessed through them. F-Stop Gallery enables the user to instantly search through all their photos to find results based on file name and metadata (tags, ratings, camera model, etc.).

It also allows the user to save tags (keywords) and ratings in XMP format so they stay with the image and can be read by popular programs such as Lightroom, Picasa, Aperture, Windows Live Photo Gallery, digiKam and many more. Further, it can read metadata directly from images (EXIF, XMP, IPTC) and custom sort images using drag and drop.

III. PROPOSED APPROACH

Each user of a gallery application is inherently unique with respect to the kind of images she would like or the way in which she would want the images to be organized. But most gallery apps, such as the ones mentioned in the previous section, do not exactly acknowledge this uniqueness and tend to provide the same "look and feel" for all users irrespective of how different they are from each other. Some of them do acknowledge this uniqueness and provide personalization options to the user using which she can organize the images according to her wishes. But the shortcoming of these is that the user has to do everything herself explicitly. i.e. it does not happen implicitly.

Also, another shortcoming is that a conventional gallery application displays the images within each folder according to the time at which the particular images are written into memory which means if the user intends to open an image which arrived at the phone days or months ago (say, a timetable image for a student in WhatsApp images folder), she will have to scroll through hundreds of images (that arrived much later) in order to get to the intended (timetable) image.

The proposed approach is to solve these shortcomings by developing a gallery application that would implicitly carry out the work of personalization by using machine learning [1] to learn the uniqueness of each user and potentially provide a different "look and feel" for each user based on their preferences as learned by the system. We choose to refer to the whole idea as "implicit personalization". We choose to implement it by introducing two mechanisms- Classification and Prioritization [2][3].

The former includes the classification of images into a number of classes whose names are input from the user using explicit prompt during the initialization phase which happens when the user first opens the application and if the user chooses not to provide explicit class names, then defaults will be fetched implicitly. Each class is displayed along with the class name just like a single folder is displayed by a conventional gallery application. This classification potentially helps the user to get to the intended image faster.

After having classified images into classes, the images in each class are prioritized. i.e. the images which are of higher priority to a particular user (like the timetable image to a student) are displayed first. So, the user wouldn't need to scroll through a whole lot of unintended images to get to the intended image. Potentially, this makes the life of the user a tad easier.

The classification can be implemented by making in-device predictions using the trained model Inception-V3 model [4] (through Google's TensorFlow libraries) [5] and then making use of the image captions that are output to group the images of a particular class name together and displaying them.

Let's say there is a person who happens to have a gallery mainly filled with images of people, dogs, cats, nature and buildings. These images are input into the inception model that classifies them into 1000 generic classes. Now, considering the class names that were input by the user (for instance, Animals and Humans), the images which were classified under various animal names such as dogs, cats are programmed to be displayed under the folder "Animals". The images of people are displayed under the folder with the name "Humans" and all of the remaining images which weren't classified under either of the class names are coded to displayed under the folder "Miscellaneous". The application has to be coded in such a way that, among all 1000 classes, whichever can be considered an animal like class "deer" are coded to be displayed under the folder "animals" [6].

The prioritization within each class can be implemented by creating a mechanism that can store values such as, the frequency with which the image is explicitly accessed, the amount of time for which the image is viewed before device sleeps or the user closes the image or switches to a different app. These values can be stored along with the image as manually added metadata. They can then be input as features (after normalizing them) to a Naive Bayes classifier [7] to obtain a label which in turn, can be converted into a priority value for the image. The priority value for an image potentially signifies how important that image is to the user of the device. The images inside a class are then sorted according to priority such that image that has more priority appears first when the user opens the class [8].

This is a case of precise predictions rather than classification of data points into classes and hence we will be considering the following regression algorithms and choosing the best among them by analyzing the distribution in each one:

- Linear regression
- Decision tree regression [9]
- Bayesian ridge regression
- Support vector regression [10]

As an example, let's consider three features namely, the frequency with which the image is explicitly accessed, the amount of time for which the image is viewed, and the average inter-access time. These features are normalized to a range of 0-10. Considering a Bayesian ridge regressor model that is trained by the following random dataset.

Access Count	Duration of Access	Average Inter-Access Time	Priority Value
10	10	10	100
9	9	9	90
8	8	8	80
7	7	7	70
6	6	6	60
5	5	5	50
4	4	4	40
3	3	3	30
2	2	2	20
1	1	1	10
0	0	0	0

This trained model will be integrated into the app and the predictions will be made on the device because predicting would need minimal computational power.

For the purpose of understanding, let's say there are only 5 images in a particular folder (say, A, B, C, D, and E) and their feature values are as shown in the following table.

Image	Frequency of explicit access	Duration of access	Average inter access time	Priority value
A	1	8	4	43.33
B	2	8	5	49.99
C	5	7	9	69.99
D	7	9	9	83.33
E	1	2	3	19.99

Therefore, the model predicts that the image D is of most importance to the user and image E is of least importance. Hence, the images will be shown in the order D, C, B, A, E with D being at the top and E being at the bottom.

IV. PROS AND CONS

The main advantage of using machine learning to classify images is that it provides personalization to the user without having to be explicitly programmed to do so. On the long run, this increases consistency which consequently enhances user experience. This will also serve as an incentive for prolonged usage of the application which will in turn benefit the developers. From a commercial perspective, this approach could substantially increase the revenue of the business that provides these services.

Further, prioritizing these classified images will considerably reduce the time required to access an intended image, which would give any application that employs these techniques, an edge over existing conventional applications.

On the other hand, employing machine learning techniques for mobile applications require more computational power than typical applications and this could get expensive over time and without sufficient data to train the model, maintaining consistency becomes an issue as well. In addition to this, it is also possible that some users are not very open to allow their data being recorded even if it is only to enhance their experience.

V. CONCLUSION

Through the proposed system, we intend to develop a customized gallery application. This application classifies the images based on explicitly specified class names which will help the user to retrieve any image faster. It will also prioritize the images which make it easier to access/ delete if needed. These features make the application more efficient and user friendly compared to other typical gallery applications.

REFERENCES

[1] Amanpreet Singh, Narina Thakur & Aakanksha Sharma, "A Review of Supervised Machine Learning Algorithms", IEEE-2016, International Conference on Computing for Sustainable Global Development (INDIACom).

[2] David Opitz & Richard Maclin, "Popular Ensemble Methods: An Empirical Study", Journal of Artificial Intelligence Research 11 (1999) 169-198.

[3] Siddhartha Sankar Nath, Jajnyaseni Kar, Girish Mishra, Sayan Chakraborty & Nilanjan Dey, "A Survey of Image Classification Methods and Techniques", 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).

[4] Xiaoling Xia, Cui Xu & Bing Nan, "Inception-v3 for Flower Classification", 2017 2nd International Conference on Image, Vision and Computing.

[5] Peter Goldsborough, "A Tour of TensorFlow", arXiv, Oct 2016.

[6] Zhongjie Li, Rao Zhang, "Object Detection and Its Implementation on Android Devices".

[7] Yuguang Huang & Lei Li, "Naive Bayes Classification Algorithm Based on Small Sample Set", Proceedings of IEEE CCIS-2011.
[8] Rui Xu, "Survey of Clustering Algorithms", IEEE Transactions on Neural Networks, Vol. 16, No. 3, May 2005.
[9] Qing-yun Dai, Chun-ping Zhang & Hao Wu, "Research of Decision Tree Classification Algorithm in Data Mining", International Journal of Database Theory and Application, Vol.9, No.5 (2016), pp.1-8.
[10] Durgesh K. Srivastava & Lekha Bhambhu, "Data Classification using Support Vector Machine", Journal of Theoretical and Applied Information Technology, 2005 - 2009 JATIT.