The University of Hong Kong
Department of Computer Science

Final report

Final year project
Image matching lost and found system

<table>
<thead>
<tr>
<th>Name</th>
<th>UID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chu King Hei</td>
<td>3035693014</td>
</tr>
<tr>
<td>Moon Lok Yin</td>
<td>3035690177</td>
</tr>
<tr>
<td>Tam Yat Hin</td>
<td>3035690189</td>
</tr>
</tbody>
</table>

Supervisor
Dr. Luo Ping
Abstract

The image matching process of current lost and found systems is done manually by the transport operator staff. This is not efficient in terms of the cost of time and manpower. The machine learning model is a useful tool that can help us perform the same task automatically. Therefore, this project explores the way to implement a web application for the lost and found system that utilize the machine learning model. The accuracy of the model is one of the most crucial parts in the project. Thus, the ResNet model design is chosen to obtain a high accuracy in the image classification task used for the application, while efficientNet-B7 feature extraction and KNN with cosine similarity distance are being used to perform the image matching task for the application. The models are successfully implemented and integrated to the application. The final product application are able to perform automatic, efficient, and accurate image matching process.
Acknowledgments

First, I would like to show my gratitude for the Department of Computer Science, and Dr. Luo Ping for giving my group many suggestions about the machine learning model design.

I also appreciate the help of my groupmates for their work and cooperation.
# Contents

Abstract i

Acknowledgments ii

List of Figures vi

List of Tables vi

Abbreviations vi

1 Introduction 1

1.1 Background and motivation . . . . . . . . . . . . . . . . . . . . . . . 1
  1.1.1 Current Lost and found situation and procedures . . . . . . . 1
  1.1.2 Flaws of Existing Lost and found system . . . . . . . . . . . . 1
  1.1.3 Machine learning models . . . . . . . . . . . . . . . . . . . . 3

1.2 Objective . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 Project contribution . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.4 Outline of the report . . . . . . . . . . . . . . . . . . . . . . . . . 4

2 Methodology 5

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2 Overall design of the integrated system . . . . . . . . . . . . . . . . 5
  2.2.1 User side workflow . . . . . . . . . . . . . . . . . . . . . . . . 5
  2.2.2 Operator side workflow . . . . . . . . . . . . . . . . . . . . . 6
  2.2.3 Role of image matching model . . . . . . . . . . . . . . . . . 7
  2.2.4 Role of image tagging model . . . . . . . . . . . . . . . . . . 7
  2.2.5 Overall Workflow . . . . . . . . . . . . . . . . . . . . . . . . 8

2.3 Frontend development . . . . . . . . . . . . . . . . . . . . . . . . . 9
  2.3.1 Programming language . . . . . . . . . . . . . . . . . . . . . 9
  2.3.2 User side application . . . . . . . . . . . . . . . . . . . . . . 10
  2.3.3 Operator side application . . . . . . . . . . . . . . . . . . . . 10

2.4 Backend development . . . . . . . . . . . . . . . . . . . . . . . . 11

2.5 Image matching . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
List of Tables

3.1 Image matching: Score Accuracy with MSE ......................... 39
3.2 Image matching: Score Accuracy with ABS ...................... 39
3.3 Image matching: Score Accuracy with COSINE ................. 40
3.4 Project schedule ...................................................... 43

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>COSINE</td>
<td>Cosine Similarity</td>
</tr>
<tr>
<td>KNN</td>
<td>K-nearest neighbor</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>ResNet</td>
<td>Residual Neural Network</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background and motivation

1.1.1 Current Lost and found situation and procedures

It is common that one may lose items while using public transport. In each month, about 7000 items were found in the Mass Transit Railway (MTR). Therefore, every transport operator has a lost and found system.

Most current lost and found systems involve the following procedures. First, the owner of the lost item uploads the description or image of the lost item to the system. Then the transport operator will receive the request and ask their staffs to look for the item. If they have found some matched items, they will notify the owner through the same system used. One can check if the lost item is within the matched items suggested by the staff.

Usually, the matching process requires lots of manpower and processing time, especially when the company is large. We will further explained this in the next section, with examples of some existing lost and found system.

1.1.2 Flaws of Existing Lost and found system

The current existing lost and found system of most of the operators requires lots of manpower and processing time. For example, the Hong Kong International Airport (HKIA)

![Fig. 1.1. The Lost and found website of Hong Kong International Airport [1]](https://www.hongkongairport.com/en/passenger-guide/airport-facilities-services/lost-and-found)
only have a manual lost and found system. As shown in the website in Figure 1.1, if one has lost item in the HKIA, the only possible way to claim his/her lost item is to make arrangement through phone call. This would make the whole process inconvenient, inefficient and time consuming.

![Feedback Form](https://www.bravobus.com.hk/onlineform/en/report-lost)

Fig. 1.2. The Lost and found website of Citybus and New World First Bus [2]

On the other hand, Citybus and New World First Bus have an online lost and found system, but instead, require the applicant to input many number of fields of information, as shown in Figure 1.2. This may reduce the time and manpower needed for collecting the information but will not be significantly helpful for the item matching process. In particular, the last field that the applicant can input is a description of the lost item with not more than 750 characters. If the applicant has input too little description, it will be difficult for the operator to search the item and verify whether the item found is belonged to the applicant. As mentioned in the last section, there are thousands of lost items found and kept by the transport operator, even if the user has input very detailed descriptions,
searching among these items is still complex, difficult, and time-consuming. Moreover, since there are usually some common types of lost item found for each operator (such as luggage for the airport), it is not unusual that many applicants have lost similar item and hence provide similar descriptions, which make the operator nearly impossible to perform item matching and verification of the item owner. 

These issues show that a well-designed, automatic lost and found system is urgently needed for enhancing efficiency of the lost and found process, in order to reduce the manpower needed, time used and the administration cost. To develop such a system, machine learning model can be used to handle the item matching process autonomously.

1.1.3 Machine learning models

In modern days, machine learning is popular in both business and research aspects. It is suggested that many human activities will be performed by AI automatically in the future [3]. This will bring us not only labor cost reduction, but also higher task performance quality and increased global productivity growth. In fact, many complicated tasks such as image classification, speech recognition and text-to-speech can already be handled by machine learning models.

In our project, image matching and classification is the task that our system needs to perform. The first one is to take an target image and a set of other images as input as input and output a set of images within the input set that have the highest similarity to the target image. The second one is to take an image of object as input, and output the object tag name within a predefined output set that contains all the possibilities of the object tag.

If the item owner and operator are able to provide an image of the lost item, then we can perform item matching easily, by using the image matching model and classification model to perform image matching automatically. This will probably bring a huge increase in the efficiency of the whole lost and found process.

1.2 Objective

In this project, we aim to implement a web application for the lost and found system to provide automatic and efficient item matching process by utilizing the machine learning models. The scope of this project includes three parts, with the basic implementation of frontend website and backend database, development of machine learning models, and the integration of the models and the application.
1.3 Project contribution

An automatic lost and found application is anticipated to be appreciated by different public transport operator such as Mass Transit Railway (MTR), the Kowloon Motor Bus Co. (KMB) and the Hong Kong International Airport since the automation will probably reduce the labor cost and the overcall time consumed in the lost and found process, making it more efficient. As a passenger, an efficient lost and found system will be helpful for them to get back the lost item quickly whenever one lost their item. Therefore, it is hoped that the application developed this project will bring benefits to both the owner and passenger of different public transports.

1.4 Outline of the report

The report is structured into four chapters. The first chapter offers an overview of the lost and found procedures. This chapters also provides a review of the current systems and their drawbacks, with a brief description about the solution to the drawbacks, machine learning model. The goals and deliverable of this project are also annotated.

Chapter two presents the methodology used in the project. The overall designs of the whole lost and found system, as well as the detail implementation design of the frontend and backend application, will be explained. Furthermore, the algorithm for building the machine learning model will also be described.

Chapter three presents the UX design and the implementation of the whole application, as well as the performance of the image matching model and the image classification model. The project schedule will also be illustrated.

Chapter four concludes the whole report. It summarizes the objective, the major contribution and the product delivered of the project.
2 Methodology

2.1 Introduction

This chapter first presents the overall workflow of the whole system. Then it presents the technology we used and implementation design for each part of the system, including the frontend application, backend application, and the machine learning model. In this chapter, user refers to the owner of the lost item while operator refers to the organization that found the lost item.

2.2 Overall design of the integrated system

2.2.1 User side workflow

Figure 2.1 shows how the user can utilize our application when he/she lost an item. When the user wants to claim a lost item, one can submit a request through the frontend application. The user will be required to upload some basic information, including the venue and the estimated lost datetime, as well as the contact information. The user can choose whether or not to provide a image of the lost item.

![User perspective diagram](image)

**Fig. 2.1.** The user side workflow
If the user has uploaded an image, the image will be passed to the image matching model immediately. The model will return a set of operator uploaded images from the database which have the highest similarity with the image input. If the user finds his/her lost item within the returned set, he/she can submit a pending request with required information and wait for the operator to verify. After submitting the request, a unique reference number will be generated. The user can use it to track the status of the request. If the request is accepted, the user can get back the lost item. If it is rejected, the user can check whether the information submitted is inaccurate. In such case, the user may consider to submit a new request with the correct information and this will start a new request cycle.

If the user cannot provide an image, he/she can choose some suitable tags of the lost item, from the set predefined by the operator. Then the application will returned the set of operator uploaded images from the database which are of tags selected. If the user finds the lost item within the returned set, he/she can again, submit a pending request with required information and wait for the operator to verify.

2.2.2 Operator side workflow

Figure 2.2 shows a summary of the operator side workflow. Whenever the operator staff find a lost item, one should upload a photo of the item to the server, with required information, including the tag, found venue, and found datetime of the lost item. The photo will be passed to the image matching model, and it will return a set of user uploaded images from the database which have the highest similarity with the image input. If no item matched is found in the returned set, the operator will wait for the pending request. If a pending request is received, the operator can verify the information and choose to accept or reject the request. If the operator choose to accept, then the operator can wait for the user to come to get back the lost item. Afterwards, the operator can confirm, through the frontend application, that the item has been returned, indicating the whole lost and found cycle of the lost item is completed. If instead the choose to reject the request, then the operator again has to wait for another pending request and perform verification.

If an item matched is found in the returned set from the image matching model, this probably means that the item owner has already submit a request with image provided. Then, the operator can send an email to the user of the image matched to confirm whether he/she is the owner of the lost item found. If the user has confirmed that he/she is the item owner, then the operator can wait for the user to come and return the lost item to the user, as well as confirming it in the application. If the user has claimed that he/she is not the item owner, then the operator should wait for pending request, and the whole process will be similar to the case that no item matched is found in the returned set from the image matching model.
2.2.3 Role of image matching model

The role of the image matching model within the application is already mentioned in section 2.2.1 and section 2.2.2. Whenever an image is uploaded to the database, it will be passed as an input to the image matching model immediately. Then a set of images in the database that are the most similar to the uploaded image will be returned. It is important to note that if the input image is uploaded by the user, then the set of output images must be the images uploaded by the operator, or vice versa.

2.2.4 Role of image tagging model

As mentioned, the operator is required to provide a tag when submitting a image. The image tagging model will take the operator uploaded image as input, and return the best suitable tag for the lost item, so that the operator does not have to search for the satiable tag hardly when there are hundreds of tags existed in the system.
2.2.5 Overall Workflow

Figure 2.3 shows the overall workflow of the application. When the user wants to retrieve the lost item, one can submit a request through the frontend application. There are four possible situations in total, which are

1. The operator found the lost item and uploaded its image to the database **before** the user submit a request **with** an image.

2. The operator found the lost item and uploaded its image to the database **after** the user submit a request **with** an image.

3. The operator found the lost item and uploaded its image to the database **before** the user submit a request **without** an image.
4. The operator found the lost item and uploaded its image to the database after the user submit a request without an image.

In the first case, since the operator has already uploaded the lost item image, when the user submit his/her image, an image match should be found in the return set from the item matching model. Then the user will submit a pending request and wait for verification, as mentioned in the user side workflow.

In the second case, since the image of the lost item is not yet uploaded to the database, when the user submit his/her image, no image match can be found in the return set from the item matching model. The user can still submit a request. When the operator has finally found the item and upload its image to the server. This time, the user uploaded image should be found in the return set of the item matching model. Then the operator will send an email to the user and confirm if he/she is the item owner, as mentioned in the operator side workflow.

In the third case, since the user has no image, he/she will choose tags and check for lost item images. If a correct tag is selected, the user should be able to find its item within the images returned since the operator has already uploaded the lost item image. Then the user will submit a pending request with required information, as mentioned in the user side workflow.

In the fourth case, the user will again choose tags and check for lost item images. This time, the user will not be able to find the lost item since the operator has not yet found the item. Then the user has to wait until the item is found and then try searching again, as mentioned in the user side workflow.

Expect for the fourth case, in all other cases, the user does not have to wait and can submit a request anytime to claim the lost item, which is convenient and efficient for the user. Especially for the second cases, after submitting the request with image, the user wait until the operator found a lost item and believed that it is the one that the user is finding. Then the user can check and confirm whether or not it is the item he/she lost, through internet, without going to the operator and check it in-person instead. This will reduce many unnecessary wasted time when there is a mismatch, making the process more efficient.

2.3 Frontend development

2.3.1 Programming language

React [4], a widely used JavaScript library created by Facebook, will also be used to build the user interface. A major library being used is Material UI [5], a library of React UI components that implements Google’s Material Design. These are some efficient, powerful, and flexible tools for building a fast, scalable, and simple interactive frontend.
web application.

### 2.3.2 User side application

Based on what discussed in section 2.2, the frontend application for the user should consist of the following pages and functionalities:

- Page for uploading and cropping the image
- Page for displaying the set of returned images from the image matching model
- Page for viewing the lost item images by choosing tags
  - Function to confirm if there is an image matched
- Page for inputting the basic information, including the venue, the estimated lost datetime, and the contact information
- Page for showing the unique reference number generated after the user submitted a request
- Page for checking the request status using the reference number
- Page navigated from the email link, used for the user to check and confirm when the operator believe that the item found is the user’s lost item.

### 2.3.3 Operator side application

Based on what discussed in section 2.2, the frontend application for the operator should consist of the following pages and functionalities:

- Page for uploading and cropping the image
- Page for displaying the set of returned images from the image matching model
  - Function to confirm if there is an image matched
- Page for inputting the basic information, including the tag, venue, found datetime of the lost item
  - Function to define and create new tag
  - Function to define and create new venue options
- Page for viewing and verifying the pending request
- Function to send email for confirmation from user
- Function to confirm that a lost item has been returned

For convenience, easy management, and flexible usage, the frontend application for the operator should also possess the following pages and functionalities:

- Page for login
- Log out function

10
• Dashboard (homepage) for
  – viewing basic statistics, including
    ∗ Number of lost item found this month
    ∗ Number of request received this month
    ∗ Total number of pending request that are not verified yet
    ∗ Top 5 hot spots and their corresponding number of lost item found
    ∗ Hot tags and their corresponding number of lost item found
  – leaving message to other staff
• Page for viewing the list of existing tags/venue, and their corresponding number of lost item found
  – Function to edit the existing tag/venue name
  – Function to delete the tag/venue that are dummy (no lost item found)
• Page for viewing the list of all found lost item images, and their information
• Page for viewing the list of all requests received
• Function to filter/sort the list of tag/venue/lost item images/requests
• responsive web design

2.4 Backend development

The backend application is mainly responsible for receiving any data from the frontend application, passing it to the machine learning model or the database, retrieving the result from the model or the data from database, and sending it back to the frontend application.

Django, a free, open-source and high-level Python web framework, is used to build the server-side application. Its middleware provides an easy communication with the database and effective management of the data.

The image matching, or tagging will be performed in the backend since it is much more fast and efficient. Whenever the frontend application needs to perform image matching/tagging, it will call the API and send the input image to the backend. Then the backend application will further pass the image received to the model and send back the return images/tag to the frontend.

Django Rest Framework, a package built on top of Django, is used to build the backend APIs. With the help it, a simple, flexible, consistent RESTful API can be built.
2.5 Image matching

2.5.1 K-Nearest Neighbors Algorithm

K Nearest Neighbor (KNN), a simple, non parametric algorithm, can be used to identify a similar image. It takes a target points and a set of other points as input. Then, for each point in the input set, the algorithm will compute the distance between this point and the target point based on the predefined distance function. The points that give smaller distance value have higher similarity to the target point. After the computation, the algorithm will output $K$ number of points inside the set that have the highest similarity to the target points, where $K$ is some positive integer.

We note that the so-called "points" can be images, vectors, matrices, or any type of things, as long as the predefined distance function takes any two points of same type and return a number that represent the distance. Then the algorithm will work. Therefore, the "points" can also be "item images". Since the items of same type should be similar, and hence the points of item images of same type should have smaller distance and nearer to each other. Therefore, we can apply the K Nearest Neighbor (KNN) algorithm to obtain a set of most similar item images.

Figure 2.4 has shown an example of the application of K Nearest Neighbor (KNN) algorithm on two categories of item. The green and orange dots represents the items of category A and category B respectively. The dots of same colour are closed to each other, meaning that items of the same category have smaller distance and hence higher similarity, which is logically true.

![Fig. 2.4. Example of how KNN works](image)

In fact, KNN is an algorithm that does not require training and the number $K$ can be adjusted any time. Thus, it is flexible and elastic to process various images in run-time
application. Nonetheless, the performance of the image matching model is very crucial for the application. If the model has a poor performance and keep matching wrong images, the time cost and manpower required may even be larger than the common lost and found system. Hence, to ensure that good model performance, we have to decide the best distance metric and the best type of "point".

2.5.2 Distance function

Although there are different types of point that will be explained in the next section, for now, just note that the points will essentially be a vector. Hence, any distance function that work for vector points will be valid. Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Cosine Similarity (COSINE) are three distance functions that are used popularly for the vectors. The following are their mathematical formulas,

\[
RMSE = \sum_i (x_i - y_i)^2
\]

\[
MAE = \sum_i |x_i - y_i|
\]

\[
COSINE = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}}
\]

where \(x_i\) represents the \(i\)-th entry of the first vector, and \(y_i\) represents the \(i\)-th entry of the second vector.

We will test for these three distance functions one by one and use the one that gives the best performance.

2.5.3 Type of point

There are four types of point that we are interested in.

The first type is the image itself. Usually, the image is stored as a pixel number matrix, but we can simply flatten it to transform it into a vector. In the remaining part of the report, plain KNN refers the KNN model that takes original images as input points.

In fact, the test for the plain KNN is mainly used for comparison only, and we do not expect it will give a good performance since there are two major issues. The first one is that it has low effectiveness. Nowadays, images usually have a large size and contain millions pixels. Computing the distance between this large number of pixels is very time-consuming. Another major issue is that the plain KNN only consider the pixelwise difference. It is not possible for both humans and AIs to decide whether the two images
are of same type only based on the pixels. The lack of spatial information will make plain KNN have a poor performance in item matching. To due with these two issues, we should instead extract the features of the images, and then treat these feature vectors as "points" to be inputs of the KNN. Since items of the same type should have similar features, it is feasible to perform image matching by using the feature vector instead of the image. This will not only reduce the input size and hence the computational cost, but also provide much more spatial information for the model to match items.

Therefore, the decision now becomes what model should be used to extract the features. We can make use of different deep learning models. Usually, a deep learning has similar structure with figure 2.5. It consist of numbers of CNN layer and also the fully-connected layer. Right before the first fully-connected layer, we have the last layer of the CNN, which is in fact all the features extracted by the whole CNN. Thus, we can first pass the image to the deep learning model, then extract the last CNN layer, treat it as the feature vector "point" and input it to the KNN. In other words, we treat the whole CNN as a feature extractor to help us generate the feature vector.

![Fig. 2.5. Structure of deep learning model and the idea to extract feature vector](image)

Residual Neural Network and EfficientNet-B7 are two of the most famous deep learning image classification model. We can use the method mentioned above to extract the feature vector and pass it to the KNN.

Another type of model that we will use is the autoencoder. Figure 2.6 shows the high-
level structure of the autoencoder. A autoencoder consists two parts, which is the en-
coder and the decoder. The encoder is to extract the feature from the image. Then, the
decoder will use the encoded feature to reconstructs the image. Then the pixel-wise
difference between the original image and the reconstructed image will contribute to
the training loss and being back-propagated to tune the parameter of the encoder and
decoder.

The reasoning behind this structure is that it is hard to evaluate the encoder performance
based on the encoded feature since it is not readable for human. However, if the image
generated by the decoder is similar to the original image, it is clear that the encoder have
a good performance on extracting meaningful, important features of the image.

Therefore, our idea is to first build train an autoencoder, then we will use the resulting
encoder block to help us for extracting the features. As shown in figure 2.7, we can first
pass the image to the pre-trained encoder, then pass the resulting feature vector as an
input point to the KNN.

This whole training process does not require any operator staff to monitor and only
complete automatically by machine learning model and algorithm. In addition, the training for both autoencoder and KNN does not require any label data. Therefore, the training cost is low for gathering datasets.

In conclude, there are four types of point we can input to the KNN. The first one is the image itself. The second one is the ResNet extracted feature. The third one is the EfficientNet-B7 extracted feature, while the last one is the autoencoder extracted feature.

2.5.4 Comparsion between models

As mentioned, there are three types of distance functions and four types of point for the KNN. Then there are 12 different image matching model to test on and compare their accuracy.

Since the final output of the model is a set of with K number of images, for effective evaluation, we need to decide a simple score function to take this set as an input and return a number to represent the performance of the models. In particular, we want to have large proportion of images in the return set that are similar to the input target image. Since images of same type are considered similar, therefore, to calculate the score, we simply divide the number of images in the return set that are of the same type with the input target image by number K, that is the total number of images in the return set.

We will evaluate on the CIFAR10 [6] dataset, which contains 60,000 color images of 10 classes, with dimension 32x32. It is separated in 50,000 training images and 10,000 testing images.

2.6 Image classification

2.6.1 ResNet model

Currently, there exists many published image classification models with good performance, such as AlexNet, VGG and ResNet. In our project, the ResNet-50 models will be adopted with some modifications if needed. The reason is that ResNet model tackle the gradient vanishing problem which many models do not. Usually, during training, the loss is back-propagated to tune and optimize the matrix parameter of the model. However, if without residual network, the upper the layer, the smaller the effect induced by the loss on the parameter of the layer. This means that the model will be very difficult to optimized and hence the accuracy is limited. This is a simple summary of the gradient vanishing problem. Because of this problem, the model accuracy is reduced rapidly as we increase the number of model layers. Therefore, Microsoft has introduced residual network [7] in order to due with the gradient vanishing problem.
Fig. 2.8. Residual learning: a building block [7]

Figure 2.8 shows the crucial structure of residual network. Within residual network, shortcut connections are provided between some of the lower layers and some of the upper layers. By performing identity mapping through these shortcuts, the data can be transferred from the upper layer to a much deeper layer. In other words, the middle layers are skipped when transferring the data or back-propagating the loss, and thus the effect induced by the gradient vanishing problem can be reduced. This is the reason why the ResNet-50 model have good performance in image classification, with high accuracy, easy optimization, and high training speed.

2.6.2 Model training

ImageNET [8] is a huge dataset that are suitable for training classification model. Nonetheless, since training with too large number of categories will result in huge amount of training time and many categories in the imageNET are not likely to be lost item, hence we extract only 98 categories that are the common lost items and use this subset as the training dataset for the image tagging model. These categories includes the notebook, water bottle, umbrella, and so on. There are more than 1300 images for each category in the dataset.

The Adaptive Moment Estimation (Adam) optimiser will be used to normalise the training data first before the model training because it helps the model to converges more effectively when using a large training dataset. After some simple testing, we have decided to use a batch size of 128, and initial learning rate of $10^{-4}$.

Moreover, the images has to be resized to dimension 224x224 in order to fit it the model. This may cause the model to have a low accuracy since the proportion of the background will usually increase when we directly resize a large image into a smaller dimension. This may make the model be overfitted to the background of the training images. It may
Fig. 2.9. The sample of data augmentation

wrongly classify the input image based its background and hence have a low accuracy. If this is the case, we will implement the learning rate decay and data augmentation one by one to reduce overfitting and improve the accuracy.

For the data augmentation, in order to increase the proportion of the object in the input training image, the image is first resized to dimension 256x256. Next, the resized image will be centre cropped to dimension 224x224. Figure 2.9 provides an example to show difference between the image after directly resizing and the image after augmentation. As shown, the proportion of the object in the image after augmentation is much higher than that of image after directly resizing.

Furthermore, random horizontal flip and random rotation will also be performed to boost the randomness of input images.

3 Result and findings

This chapter presents the result and findings of the project. First, it showcase the delivered product application and explain the UI design. After that, the resulting performance of the image matching model and image tagging model will be discussed.
3.1 Application

The application has been implemented based on the discussion in section 2.2. We will first demonstrate the application for user, that is the item owner. Afterwards, the application for the operator will also be presented.

3.1.1 User side application

Figure 3.1 has shown the homepage implemented for the user that want to find the lost item. In the homepage, there are two services that the user can opt to. The first one is to submit request. In this case, user can click on the purple button, as shown in figure 3.1. Another service that the user can choose is to check the status of the submitted request. To do so, the user can click on the blue button below in the homepage.

![Fig. 3.1. User side: Home page](image)

If the user has chosen to submit a request by clicking the purple button, then the application will ask whether or not the user have the image of the lost item, as shown in Figure 3.2.
**Fig. 3.2.** User side: Page asking if the user have an image to submit

If the user have the lost item image, then the user can click the green "YES" button in figure 3.2. The application will navigate to the page in figure 3.3.

**Fig. 3.3.** User side: Page for user to upload an image

At the top of this page, a simple progress bar can be found. It shows the steps required for the user to successfully submit a request. The finished and current step the user in will be highlighted in blue so that the user can follow the step easily. As shown in the bar, the first step is to upload the image. The user click on the "SELECT" button below the progress bar and then choose and upload the image from his/her device.

After uploaded the image, the application will ask the user to crop the image in the page in figure 3.4.
Once the user click on the green "CROP" button at the bottom of the page, the application will crop the image, call the backend API with the cropped image. The backend will perform image matching with the list of images uploaded by the operator in the database. Then the backend will send a response to the frontend with the return set from the model, and the images inside will be displayed in the page in figure 3.5.

These displayed images are the similar image in the database uploaded by the operator, from the most similar in the top left to the least similar in the bottom right. If the user think that one of the image is the item he/she has lost, he/she can click on the image. For example, in figure 3.5, the first image in the top left corner looks similar to the lost item image submitted by the user, then the user can click on it. Then in figure 3.6, a panel is shown up to let the user has a more full, detailed look of the image and the user can click "YES" to confirm that it is the item he/she has lost.
Afterwards, the application will ask the user to input the venue, estimated lost datetime and the contact information, as shown in figure 3.7a. In particular, the lost venue is chosen from a list predefined by the operator. If the venue is not in the list, it means that there is no item found in that venue. Moreover, if the user have travelled more than one place and does not sure where he/she lost the item, he/she can provide multiple venues, as shown in figure 3.7b.

After inputting the information, the user can click the "COMPLETE" button and the request will be submitted and sent to the operator. If a success response is received from the server, the application will navigate to the status page of the submitted request and a success message will be shown up in the top right corner, as shown in figure 3.8. This page also shows the newly generated unique reference number for the submitted request. The user can keep it so that he/she can track the status from time to time to see if the operator has verified the request.
Fig. 3.8. User side: Successfully Submission of request and reference number

To check the status by reference number, the user can go back to the homepage in figure 3.1 and click the blue bottom below. Then as shown in figure 3.9, the application will ask the user to input the reference number.

Fig. 3.9. User Side: Page for request status checking

After inputting the reference number and clicked the "CHECK" button, the status will be displayed in the page in figure 3.10. There are four possible status that the user will see. The "Pending" status means that the request is not yet verified by the operator. The "Accept" status means that the operator has approved the request but the item has not been returned to the owner. The "Returned" status means that the operator has approved the request and the item has been returned to the owner. At last, the "Reject" status means that the operator has rejected the request.
If the user unfortunately do not find the lost item in the item matching page, just like the example in figure 3.11. Then the user can click on the "MORE" button to check more images from the database, as shown in figure 3.11b. If the user still cannot find his/her item within the images, he/she can click the "I DO NOT FOUND MY ITEM" at the bottom. Then the application will ask the user to input information, similar to the case where user has found the lost item within the images.

The only difference is that after submitting request, the user has to wait an email from the operator. When the operator find an item and believe that the item is owned by
the user based on the image submitted, then an email to is sent to the user to ask for confirmation. Figure 3.12a shows an email that is received by the user. In the email, there is a hyperlink that will direct the user to the page in figure 3.12b to confirm whether or not he/she is the owner.

![Email received](image1)
![Confirmation](image2)
![After confirm "YES"](image3)

**Fig. 3.12.** User side: Email received and claiming the lost item

If the user confirm that he/she owns the item by clicking the "YES" button, the status of the request will be changed to "Accept", since the operator has already agreed that the lost item is owned by the user. Then the user can find the operator and get back the lost item.

If the user cannot provide any lost item image, he/she can click "NO" in the page in figure 3.2 and the application will navigate to the page where the user can search image by choosing one or multiple tags, as shown in the page in figure 3.13.
After selecting the tags, the application will display all the images uploaded by the operator that are of one of the selected tags. If the user finds the lost item within the filtered images, he/she can select the image and confirm that he/she owns the item. The remaining steps are similar to the case where the user finds a match when submitting image. The user will have to provide info and submit request through the page in figure 3.7a, then get the reference number to track the request status occasionally.

If the user cannot find the lost item within the filtered images, this means the operator has not yet found the lost item. The user may wait and try again later.

### 3.1.2 Operator side application

In any page of the user side application, a login button is placed in the top right hand corner. Clicking on it will direct the operator staff to the login page in figure 3.14.

After inputting the username and password, the staff can login by clicking the "SIGN IN" button. After signing in, operator staff can log out by clicking the button at the
top right corner again. After logging in, a bar containing 5 icons also appears in the leftmost. The staff can go to different pages by clicking the icons. The first icon is the dashboard. The second icon is the page of tag list and venue list. The third icon is the page for uploading image of lost item. The fourth icon is the page of list of lost item found. The last icon is the page of request list. These pages will be explained one by one.

The dashboard page, in figure 3.15, is the the homepage of the operator side application. It is displayed once the staff login the system.

![Fig. 3.15. Operator Side: Dashboard/Homepage](image)

As mentioned in section 2.3.3, some basic statistics can be viewed in the dashboard, with number of item found and number of request received this month in the top right hand corner, total number of pending request in the left hand side, the hot spots and corresponding number of lost item in the center, the hot tags and corresponding number of lost item in the right hand side. At last, the staff can leave message through the bottom left panel so that he/she can communicate with other staffs easily.

Figure 3.16 shows the page of staff can view all the existing tags or venues that were created, and also their corresponding number of lost item found. Both list are sorted, by default, in descending order of the number of lost item found.
If the operator staff find out that the tag name or venue name is not precise, the staff can edit it very easily by following the figure 3.17. First, the staff can click on the cell of the imprecise name, then change it by input a new name. Afterwards, the staff can press "Enter" and the application will call the backend API and send a request for changing the venue name. If a success response is received by the frontend, a success message is popped up in the top right corner to notice the staff that the name is successfully changed in database. For example, in figure 3.17, the venue name "CYM" is edited into a more detailed name called "Chong Yuet Ming Amenities Centre". The tag name can also edited in a similar manner.
If some of tag names and venue names have no lost item found, then this names are dummy. Figure 3.18 has shown how to view the dummy names easily. First, move the cursor to the trash button of the list and an arrow will appear. Then click on the arrow and all the dummy names will be sorted at the top of the list, as shown in figure 3.18b.
The staff can choose to delete the dummy names by following the step in figure 3.19. First, the staff can select the dummy name that are going to be deleted by checking the selection checkbox in the left of the list. Next, the staff can click the trash icon in the top right of the list. Then the application will ask for confirmation. By clicking "YES" to confirm, the system will call the backend API to delete the venue name in the database. If a success response is received by the frontend, a success message is popped up in the top right corner to notice the staff that the name is successfully deleted in database. For example, in figure 3.19, the dummy venue name "Eliot Hall", "Graduate House" and "Haking Wong Building" have been successfully deleted. Dummy tag name can also be deleted in a similar manner.
Another page of the operator side application is the page for uploading image, which is shown in figure 3.20a. The staff can upload the image from their device by clicking the "SELECT" button in the page. Then the application allow the staff to crop the image, as demonstrated in figure 3.20b.
Once the staff click the "CROP" button, the application will ask the staff to fill in the form in figure 3.21, to provide the basic information, including the found datetime, found venue and the tags for the lost item.

Before the user to fill in the form, a tag has already been filled in the "tag" field of the form. This tag is generated by the image tagging model and performed when the "CROP" button is clicked. When the staff clicked "CROP", the application will crop the image, call the backend API and send the cropped image to the server at the same time. The backend will receive the image and use it as an input to perform image tagging. Then the backend will send a response to the frontend with the return tag from the model. Once the response is received by the frontend, the return tag will be filled into the form automatically. For example, in figure 3.21a, the tag "water bottle" is returned from the image tagging model and filled by the application.

To input more tags and the found venue, the staff can choose from the existing tag list and venue list respectively, as demonstrated in figure 3.22.

Fig. 3.20. Operator side: Upload image

Fig. 3.21. Operator side: Input information of found item
If there are no suitable existing tags/venue to input, the staff can simply create a new one by entering the name in the field and press "Enter". Then the application will call the backend API to create a new tag/venue name in the database. Once the frontend receives the response, a success message will be popped up and the new name is added to the list of selection. For example, figure 3.23 has demonstrated how to create the tag name "green" and the venue name "Knowles Building".
After filling the form, the staff can click the "SUBMIT" button in the bottom. Then the image will be sent to the server and stored in the database. At the same time, the backend will filter the images in the database uploaded by the user using the found venue and found datetime that the staff has just submitted. The the image matching model is used to sort the filtered set in descending order of the similarity with the image just received from the frontend. Afterwards, the return set will be sent back to the frontend. Once the frontend has received, it will do the checking. If the set is empty, then it navigate back to the page in figure 3.20a. Else, the application will display the return set in the same order from left to right and from up to down, as shown in figure 3.24a.
Then the staff can check, if there is an image that look similar to the staff uploaded one, then it probably means that the item owner has realized that he/she has lost an item and have submitted the request. In this case, the staff can click on that image to have a more detailed look on the image. If the staff believe the two item are the same, then the staff can click the green button "EMAIL TO THE REQUESTOR", as shown in figure 3.24b. Now, a email will be sent to the user to ask him/her to confirm whether or not he/she is the item owner. If the email is sent successfully, a message will be popped up to notice the staff, which is demonstrated in figure 3.24c. The whole image uploading cycle for the operator is completed.

The page in figure 3.25a is the page of lost items. In this page, initially, the staff can check all the lost item that have been found. By clicking on any of these items, the staff can check the details info about the clicked lost item, as presented in figure 3.25b.
The staff can filter the list of lost item by providing the date of found, or the found venue, or the item tag(s). For example, in figure 3.26a, after filtering, all the "mouses" and "phones" found on "14/4/2023" in the "Main Library" are displayed. Moreover, the staff can also filter the images by two classes. The first class is the "INVENTORY" class, which are the lost items that have not yet been returned to the owner, while the other class is the "SEND" class, which are the lost items that have already been returned to the owner. An example of filtering the images by the two classes is presented by Figures 3.26b and 3.26c.
The last page is the request page in figure 3.27. In this page, the staff can view all the requests that have been received, and also their information, including the submission date, user contact info, and the status of the request. Each request row has an info button in the end. If the staff click on the info button, the application will navigate to the verification page of the request of that row.

![Request List Page](image)

**Fig. 3.27. Operator side: Page of request list**

Figure 3.28 shows the interface of the verification page. In the verification page, it will display the information including the image (if any) submitted both by the user and by the staff, so that the staff can compare the difference to decide whether or not to approve the request. Our system will automatically detect if there is any mismatch. A warning message will be displayed in the top of the page and the mismatch information will be highlighted in orange so that the staff can be careful and recognize the difference easily. For example, in figure 3.28a, there is no mismatch and hence no warning message is shown. The staff should probably approve the request. On the other hand, in figure 3.28b, there are two mismatches and hence a warning message appears to warn the staff that there is mismatch in the lost and found datetime and the lost venue. Then the staff should probably reject the request.
The staff can accept or reject the request by clicking the button at the bottom. Then the application will immediately call the backend API to change the status of request in the database. If a success response is received by the frontend, a success message is popped up and the status in the verification page will be changed correspondingly, as demonstrated in figures 3.29a and 3.29b.

If the request is accepted, then a new button will appear at the bottom of the verification page for confirming that the item is returned to the owner. The staff can click it after returning the item, and again the application will call the backend API to change the
status of request in the database. Once a success response is received, a success message is popped up and the status in the page will be changed to "Returned", as shown in the figure 3.28.

### 3.2 Image matching

By training the autoencoder with 50000 training images and evaluating all the models on the 10000 testing images of CIFAR10, we obtain the results in table 3.1, 3.2 and 3.3.

**Table 3.1.** Image matching: Score Accuracy with MSE

<table>
<thead>
<tr>
<th>K=</th>
<th>Plain KNN</th>
<th>ResNet50</th>
<th>EfficientNet-B7</th>
<th>Autoencoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27.01%</td>
<td>80.71%</td>
<td>85.63%</td>
<td>37.50%</td>
</tr>
<tr>
<td>2</td>
<td>25.47%</td>
<td>79.49%</td>
<td>84.94%</td>
<td>34.91%</td>
</tr>
<tr>
<td>3</td>
<td>24.79%</td>
<td>78.67%</td>
<td>84.46%</td>
<td>33.32%</td>
</tr>
<tr>
<td>4</td>
<td>24.36%</td>
<td>78.08%</td>
<td>84.06%</td>
<td>31.97%</td>
</tr>
<tr>
<td>5</td>
<td>23.93%</td>
<td>77.54%</td>
<td>83.71%</td>
<td>31.10%</td>
</tr>
<tr>
<td>6</td>
<td>23.64%</td>
<td>77.00%</td>
<td>83.46%</td>
<td>30.39%</td>
</tr>
<tr>
<td>7</td>
<td>23.31%</td>
<td>76.57%</td>
<td>83.22%</td>
<td>29.83%</td>
</tr>
<tr>
<td>8</td>
<td>23.06%</td>
<td>76.15%</td>
<td>82.99%</td>
<td>29.41%</td>
</tr>
<tr>
<td>9</td>
<td>22.82%</td>
<td>75.75%</td>
<td>82.74%</td>
<td>29.03%</td>
</tr>
<tr>
<td>10</td>
<td>22.59%</td>
<td>75.45%</td>
<td>82.54%</td>
<td>28.81%</td>
</tr>
</tbody>
</table>

**Table 3.2.** Image matching: Score Accuracy with ABS

<table>
<thead>
<tr>
<th>K=</th>
<th>Plain KNN</th>
<th>ResNet50</th>
<th>EfficientNet-B7</th>
<th>Autoencoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29.04%</td>
<td>78.37%</td>
<td>86.09%</td>
<td>37.03%</td>
</tr>
<tr>
<td>2</td>
<td>28.10%</td>
<td>76.73%</td>
<td>85.22%</td>
<td>34.24%</td>
</tr>
<tr>
<td>3</td>
<td>27.07%</td>
<td>75.66%</td>
<td>84.74%</td>
<td>32.44%</td>
</tr>
<tr>
<td>4</td>
<td>26.39%</td>
<td>74.87%</td>
<td>84.42%</td>
<td>31.26%</td>
</tr>
<tr>
<td>5</td>
<td>25.95%</td>
<td>74.07%</td>
<td>84.18%</td>
<td>30.49%</td>
</tr>
<tr>
<td>6</td>
<td>25.56%</td>
<td>73.39%</td>
<td>83.93%</td>
<td>29.81%</td>
</tr>
<tr>
<td>7</td>
<td>25.15%</td>
<td>72.70%</td>
<td>83.66%</td>
<td>29.26%</td>
</tr>
<tr>
<td>8</td>
<td>24.86%</td>
<td>72.15%</td>
<td>83.50%</td>
<td>28.74%</td>
</tr>
<tr>
<td>9</td>
<td>24.65%</td>
<td>71.70%</td>
<td>83.36%</td>
<td>28.36%</td>
</tr>
<tr>
<td>10</td>
<td>24.40%</td>
<td>71.19%</td>
<td>83.15%</td>
<td>28.08%</td>
</tr>
</tbody>
</table>
Table 3.3. Image matching: Score Accuracy with COSINE

<table>
<thead>
<tr>
<th>K=</th>
<th>Plain KNN</th>
<th>ResNet50</th>
<th>EfficientNet-B7</th>
<th>Autoencoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28.72%</td>
<td>80.46%</td>
<td>85.92%</td>
<td>48.30%</td>
</tr>
<tr>
<td>2</td>
<td>27.26%</td>
<td>79.44%</td>
<td>85.08%</td>
<td>45.08%</td>
</tr>
<tr>
<td>3</td>
<td>26.37%</td>
<td>78.52%</td>
<td>84.68%</td>
<td>43.33%</td>
</tr>
<tr>
<td>4</td>
<td>25.80%</td>
<td>77.76%</td>
<td>84.10%</td>
<td>42.08%</td>
</tr>
<tr>
<td>5</td>
<td>25.28%</td>
<td>77.24%</td>
<td>83.80%</td>
<td>41.09%</td>
</tr>
<tr>
<td>6</td>
<td>24.85%</td>
<td>76.76%</td>
<td>83.42%</td>
<td>40.24%</td>
</tr>
<tr>
<td>7</td>
<td>24.38%</td>
<td>76.28%</td>
<td>83.11%</td>
<td>39.60%</td>
</tr>
<tr>
<td>8</td>
<td>24.12%</td>
<td>75.84%</td>
<td>82.78%</td>
<td>39.10%</td>
</tr>
<tr>
<td>9</td>
<td>23.85%</td>
<td>75.42%</td>
<td>82.51%</td>
<td>38.53%</td>
</tr>
<tr>
<td>10</td>
<td>23.61%</td>
<td>75.07%</td>
<td>82.28%</td>
<td>38.09%</td>
</tr>
</tbody>
</table>

In terms of type of feature vector, the score accuracy of the plain KNN is the lowest, which is expected due to its limitations to the spatial information. The accuracy of the autoencoder is the second lowest and its score accuracy is poor. On the other hand, the score accuracy of the model with ResNet feature and EfficientNet-B7 is much higher, with EfficientNet-B7 being the highest in general. It has higher than 80% for all number of K, which is satisfactory.

In terms of distance metric, the score accuracy with cosine similarity are highest in general. Therefore, the cosine similarity distance will be used for the runtime image matching model.

3.3 Image classification

As mentioned the image has to be resize to dimension 224x224 so that it can be inputted into the model. We will experiment for directly resizing, implementing learning rate decay, and apply data augmentation to see if the accuracy will be increased.

3.3.1 Direct resize

Direct resizing of the images is first being tested and the model has been heavily over-fitted after 12 epochs, which is as expected. Thus, learning rate decay is then used and tested.

3.3.2 Learning rate decay

After implementing learning rate decay, it means the learning rate is reduced as the training progresses, and figure 3.30 provide a summary of the training and verification accuracy of the models trained with learning rate decay and the original model.
Fig. 3.30. Accuracy of classification model with learning decay but without data augmentation

The model with decay rate 0.0125 has the highest top-1 verification accuracy, which is 51.73%. However, there is only 1.20% increase, compared to the verification accuracy of the model without learning rate decay, which is 50.53%. The accuracy is still undesirable, and hence we then test for data augmentation.

3.3.3 Data augmentation

After applying data augmentation, the proportion of objects in training images is increased, and figure ?? provide a summary of the training and verification accuracy of models with data augmentation under different learning decay. The model with decay rate 0.0125 still has the highest top-1 verification accuracy, but this time it has a fair increase, from 51.73% to 58.57%. The accuracy is just fair and in the future, we will seek for some alternatives to further boost the accuracy.
3.4 Project Schedule

Table 3.4 covers the project schedule. All the project deliverable, including the project web page, whole application and the final report, have been completed.
### Table 3.4. Project schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Details</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Oct 2022</td>
<td>Completion of Detailed Project Plan and Project Web Page</td>
<td>Completed</td>
</tr>
<tr>
<td>2 Oct 2022 to 30 Dec 2022</td>
<td>Implementation of the image matching machine learning model</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Analysis on the performance of model by performance metrics</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Modification and tuning of the model based on the performance metrics</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Implementation of the backend database for storing the user request and the record of received items</td>
<td>Completed</td>
</tr>
<tr>
<td>9 Jan 2023 to 22 Jan 2023</td>
<td>Completion of preliminary implementation and detailed interim report</td>
<td>Completed</td>
</tr>
<tr>
<td>23 Jan 2023 to 31 Jan 2023</td>
<td>Preparation on the first presentation</td>
<td>Completed</td>
</tr>
<tr>
<td>1 Feb 2023</td>
<td>First presentation</td>
<td>Completed</td>
</tr>
<tr>
<td>23 Jan 2023 to 10 Apr 2023</td>
<td>Application for users to submit and tracking the application status</td>
<td>Completed</td>
</tr>
<tr>
<td>10 Apr 2023</td>
<td>Application for the operator to monitor and view the application details and the status of lost items</td>
<td>Completed</td>
</tr>
<tr>
<td>10 Apr 2023 to 16 Apr 2023</td>
<td>Completion of finalized tested implementation and Final Report</td>
<td>Completed</td>
</tr>
<tr>
<td>16 Apr 2023</td>
<td>Preparation of Final Presentation</td>
<td>Completed</td>
</tr>
<tr>
<td>17 Apr 2023</td>
<td>Final presentation</td>
<td>Completed</td>
</tr>
<tr>
<td>18 Apr 2023 to 2 May 2023</td>
<td>Preparation of Project Exhibition including poster and a 3-min video</td>
<td></td>
</tr>
<tr>
<td>3 May 2023</td>
<td>Project Exhibition</td>
<td></td>
</tr>
</tbody>
</table>
4 Conclusion

Since the image matching process in current lost and found system is inefficient in terms of both labor cost and time cost, the machine learning model can be helpful if it can handle the process automatically. Our project aims to implement a web application to handle most part of the lost and found process automatically by applying the machine learning technology.

The image matching and tagging models have been implemented and integrated with the application. A user-friendly, convenient and efficient application has been successfully implemented to handle the image matching task automatically. It is hoped that the application can be adopted by some public transport operator and hence the passengers can benefit from the efficient lost and found system whenever they lost their items. In the future, the project team wish to further improve the accuracy of the two models.
References


