The University of Hong Kong
Department of Computer Science

Interim report

Final year project
Image matching lost and found system

Name          UID
Chu King Hei  3035693014
Moon Lok Yin  3035690177
Tam Yat Hin   3035690189

Supervisor
Dr. Luo Ping
## Contents

1 **Background** 2  
   1.1 Situation of lost and found in Hong Kong 2  

2 **Objective** 2  

3 **Methodology** 2  
   3.1 Application design 2  
   3.2 Image identifying process 3  
      3.2.1 Image matching 3  
      3.2.2 Image features extraction 4  
   3.3 Image classification: ResNet 5  

4 **Current progress** 6  
   4.1 Image matching 6  
      4.1.1 Evaluation of KNN on MNIST 6  
      4.1.2 Evaluation KNN on CIFAR10 8  
      4.1.3 Evaluation KNN with AutoEncoder 9  
   4.2 Image classification 13  
      4.2.1 Dataset 13  
      4.2.2 Model training 13  
      4.2.3 Learning rate decay 13  
      4.2.4 Data augmentation 14  

5 **Project Schedule** 16  

6 **Future Plan** 17  
   6.1 Online learning for image classification model 17  
   6.2 Application 18
1 Background

1.1 Situation of lost and found in Hong Kong

It is common to find a lost item in public transport and common facilities. Mass Transit Railway (MTR) found more than 7 thousand lost items per month and 5.6 thousand cases were reported but only 30 percent to 40 percent were successfully found. Assume each case takes around 5 minutes to process, it costs 262,500 HKD to deal with 84 thousand cases per year according to the minimum wage at 37.5 HKD per hours. Since the description of the object outlook varies from person to person, only using text or verbally describing the outlook may not be sufficient to describe the object outlook. This raises the complexity for the identification process and the manpower. Therefore, a comprehensive, systematic and automatic lost and found application brings a solution to reduce the administration cost, increase the return rate of lost item and allow handling reported case efficiently.

2 Objective

In this project, we aim to implement an application for lost and found system to provide automatic and efficient process. We will attempt to integrate a machine learning model into the application so that the item matching process can be handled by the model automatically.

3 Methodology

3.1 Application design

In the original plan, when the user submits a request and uploads an image, the photo will be passed to run the image identifying process immediately. If a matching is found in the database of lost items, the user will be notified. Otherwise, the application will restart the identifying process without the image. If negative result is obtained, the user will have to submit another request again later. On the other hand, when the transport operator upload an image, there is no image matching process.

After more discussions, we decide to modify the design. When no matching is found in the first identifying process, the image submitted by the user will be stored in the database if the user agreed. When the transport operator upload an image, the application will also run the image identifying process to see if there is any matching with the images submitted by different users. If a matching is found, the user will be notified. With this design, the user is notified immediately when any new lost item image is matched. There is no need for user to submit a new request.
3.2 Image identifying process

3.2.1 Image matching

To identify a similar image, K Nearest Neighbor (KNN) which is a non-parametric is used. In Figure 3.1, the green and orange dots show the item which belongs to category A or category B. Since the item with the same category is neighboring each other, calculating the distance of the new data point between each point in the database allows finding the first k closest neighboring point. Ultimately, it could determine the category of the new data point.

Fig. 3.1. Example of how KNN works

To apply this model in the application, before processing, each image is stored as a feature vector. Then, the distance between each of those feature vectors and an input image vector is calculated. The k nearest vector will be output as the most similar image. Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are two choices for distance formula. In the following formula, $x^i$ denotes each pixel on an image stored in the database, and $y^i$ denotes each pixel on an input image from the user.

\[
RMSE = \sum (x^i - y^i)^2
\]

\[
MAE = \sum |x^i - y^i|
\]

RMSE penalizing larger error and scale with square while MAE only scale with linear. It means RMSE is more favorable in images with a lower pixel-wise difference. Further evaluation is needed to determine which formula should be applied.

This model does not require training and the k number could be adjusted at any time. Therefore, it gives high flexibility and elasticity to processing various types of lost items. However, the model has a penalty in a large set of input images.
3.2.2 Image features extraction

Nowadays, a signal image contains millions of pixels which increases the computation cost of KNN. But human does not classify an image by looking at every pixel on an image. Before doing the comparison, the shape, size, color, and structure of an image are more important to determine the difference between images. Therefore, before feeding the image to the KNN, features should be extracted. Thus, the Autoencoder brings an effective solution for feature extraction. Fig 3.2 shows encoder captures the image pixel and compresses it to an encoded feature. Then, the decoder reconstructs the image from the encoded feature. The training loss is calculated based on a pixel-wise difference between the original image and the inference image.

![Fig. 3.2. Training process of Autoencoder](image)

Evaluating the performance of the encoder is difficult because the encoded feature is only readable to the computer. However, if the decoder generates the original image with no difference from the encoded feature, it represents a meaningful and effective compression for the encoder. Figure 3.3 shows the complete process. Before feeding an image to KNN, the pre-trained encoder extract image to the encoded feature. Eventually, the k similar image reports to the user.

![Fig. 3.3. Architecture of whole image matching processing](image)

This whole 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.

### 3.3 Image classification: ResNet

Much research on image classification have been done. Many models, such as AlexNet, VGG, and ResNet, have been published. In our project, we will adopt the ResNet-50 models and make modifications on it when needed. Most models without a residual network will suffer from gradient vanishing problem. Ma model optimizes the matrix parameters by back-propagating the loss obtained. However, the effect of the loss on the parameter becomes smaller and smaller when it is back-propagated to upper and upper layer. Eventually, the loss will vanish and have no effect on the parameters of the upper layers. The model cannot be optimized and thus will have limited accuracy. This problem is known as the gradient vanishing problem and will reduce the accuracy quickly when the number of layers is increased. To tackle this problem, residual network is introduced by Microsoft [1].

![Residual learning: a building block](image)

The key structure of residual network is shown in Figure 3.4. The idea is to provide a shortcut connection between some of the layers. The data of the upper layer will be transferred directly to a much deeper layer using identity mapping, and skipping all Figure 3.4 Residual learning: a building block [1] the middle layers. In this way, the loss can be back-propagated directly to the upper layer and hence reduce the effect of gradient vanishing problem. This makes the model ResNet-50 have a high accuracy in image classification, be easy to be optimized, and have a high training speed.

The classification model is used to generated a category tag on each submitted image (by both operators and lost item owners). Before adding the image to the KNN matching model, the tag is used to filter the image in database, boast the efficient of image matching processing by reducing the number of input images. In addition, this ensure the item owner who do not have an image of the item can search the target lost item
based on the filtering result of category tag.

4 Current progress

This chapter outlines the current progress. It evaluates the model performance and shows the progression of the project.

4.1 Image matching

4.1.1 Evaluation of KNN on MNIST

To evaluate the performance of KNN, the MNIST dataset is used. [2] This dataset contains 70,000 handwritten digits images which separate into 60,000 training images and 10,000 testing images with 28x28 pixels.

Table 4.1 shows the accuracy of KNN using RMSE and MAE in 10,000 testing images.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN with RMSE</td>
<td>96.91%</td>
</tr>
<tr>
<td>KNN with MAE</td>
<td>96.31%</td>
</tr>
</tbody>
</table>

Both formulas have decent performance in classifying the handwritten digits. The RMSE is slightly better than MAE. Since there are only 784 pixels on a signal image with black color and the shape of a handwritten digit are similar, it yields a great result. Figure 4.1 shows 4 accurate results and figure 4.2 shows 4 inaccurate results. "Label" is label data from testing data and prediction is the most similar image from training data.
KNN successfully recognizes the shape of a digit. For instance, in figure 4.1, although the number "6" in testing data has a tiny stroke, the model predicts it as the number "6". However, in figure 4.2, it predicts the number "3" as a "5" because both have a reflected "c" at the bottom of an image. It seems the model does not recognize high-level differences like content and style discrepancies.
4.1.2 Evaluation KNN on CIFAR10

CIFAR10 contains 60,000 32x32 color images in 10 classes. It separates 50,000 training images and 10,000 testing images.

Table 4.2 shows the accuracy of KNN using RMSE and MAE in 10,000 testing images.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN with RMSE</td>
<td>35.39%</td>
</tr>
<tr>
<td>KNN with MAE</td>
<td>38.59%</td>
</tr>
</tbody>
</table>

Compared to the result of MNIST in Table 4.1, both formulas are underperform because of the color involved. MAE has higher accuracy than RMSE which means MAE is more accurate on recognize larger and color image. Figure 4.3 shows 4 results with 2 accurate predictions.

In figure 4.3, the model predicts the ship correctly because the background for both images looks the same that contains a white sky, red ship, and blue sea. The model does not focus on recognizing the main object such as the ship. Therefore, the model predicts the plane, not the ship because the background looks similar. To sum up, if the items do not fit in the majority of pixels of an image, it yields poor results.

From those evaluations, it seems feature extraction is vital because not every pixel on an image contains meaningful information for classifying an image. The implementation of the autoencoder is in progress and those evaluations are the baseline model of the
final model.

4.1.3 Evaluation KNN with AutoEncoder

Two AutoEncoder models with different encoded image sizes are trained on CIFAR10. Both models receive an input image with a size of 32x32 color pixels and compress it from 3*32*32 (3072) feature points to 128 and 1152 feature points separately. Table 4.3 shows the mean square error loss between the input image and the recovered image.

<table>
<thead>
<tr>
<th>Model</th>
<th>Encode image size (feature points)</th>
<th>Testing Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoEncoder A</td>
<td>128</td>
<td>0.004862</td>
</tr>
<tr>
<td>AutoEncoder B</td>
<td>1152</td>
<td>0.004178</td>
</tr>
</tbody>
</table>

AutoEncoder A has a higher loss than AutoEncoder B. However, AutoEncoder A compresses the input image 24 times while AutoEncoder B only compresses the input image 2.7 times.

In Figure 4.4 and 4.5, the encoded input image feed to the KNN and finds the most similar K images by the shortest distance. If at least one of the classes in k images is equal to the label class, it counts as success in recognition.

![Accuracy of KNN in different K values with RMSE in CIFAR10](image)

**Fig. 4.4.** The accuracy of KNN in different K values with RMSE in CIFAR10
If apply RMSE distance formula, having an encoder performs better than plain KNN. Both Encoder A and Encoder B increase the accuracy when k is equal to 1, 5, and 10. But when applying MAE distance formula, having an encoder worse than plain KNN. The possible reason is that MAE is more favorable to a larger dimension. By comparison in plain KNN, the accuracy of MAE is higher than RMSE.

Figure 4.6 and 4.7 show the top 4 similar images to the test image. The left image is the most similar and the right is less similar. The top image is the test image with a label.

Fig. 4.5. The accuracy of KNN in different K values with MAE in CIFAR10
**Fig. 4.6.** Top 4 similar images with plain KNN

**Fig. 4.7.** Top 4 similar images with Encoder + KNN
Although KNN inference the correct result of bird in figure 4.6. It may come up with the answer from the sky background, not the main content. On the other hand, in figure 4.7, a KNN successfully inference the correct answer because the encoder extracts the features like the shape of a bird. It shows an encoder determines similar images in a more perceptual way.

Overall, The plain KNN with MAE seems to yield the best results. Nevertheless, a common lost item image may contain a million feature points but not 3072 feature points like the data in CIFAR10. The encoder-A reduces the image size 24 times but only slightly affects the accuracy. Moreover, RMSE is more suitable for encoded images with a lower dimension. Therefore, an encoder with KNN using the RMSE distance formula is the most applicable to the application.
4.2 Image classification

4.2.1 Dataset
We have extracted 98 categories of commonly lost items from the imageNET[4] dataset, including laptops, pencil cases, sunglasses, and so on for classification, each category contains 1,300 images.

4.2.2 Model training
Compared to the traditional Stochastic Gradient Descent (SGD), using Adaptive Moment Estimation (Adam) optimiser and normalise the image before training to ensure the model is able convergence effectively even with a large dataset, which is crucial for applying the model for online learning. After multiple trials, we have selected a batch size of 128, with an initial learning rate $10^{-4}$.

4.2.3 Learning rate decay
At first, we directly resize the images in the dataset to 224x224 as the input of the model, and as a result, the model has significantly overfitted to the background of the image after 12 epochs.
To mitigate overfitting, we implement learning rate decay to reduce the learning rate as the training progresses. Figure 4.8 summarize the test and verification accuracy of the model. The top-1 verification accuracy slightly increases 1.20% from 50.53% on a model without learning rate decay to 51.73% with a learning rate decay of 0.0125. However, the verification accuracy is still undesirable, and we decide to work on inspecting input images.
4.2.4 Data augmentation

After a walk-through with the input images, we discover instead of the target lost item, the background covers a large proportion of the image and the model misclassified the image based on the background of the image when we directly resize the image to 224x224 (Fig. 4.9). Therefore, we decide to complete a data augmentation aim to increase the proportion of the target lost items in the input image and reduce overfitting. To increase the proportion of the items, we will first resize the image to 256x256, then centre crop the image to 224x224 (Fig. 4.9). In addition, we also implement random flip horizontally and random rotation to increase the randomness of input images.

Figure 4.10 summarize the accuracy of models after adding data augmentation under different learning decay. The top-1 verification accuracy has a large, 5.845 increase from 51.73% on the model with a learning rate decay of 0.0125 and no augmentation to 58.57% with learning rate decay of 0.0125 and augmentation. We will continue to seek out some methodologies to further improve the verification accuracy of the model.
Fig. 4.9. The sample of data augmentation

Fig. 4.10. Accuracy of classification model with data augmentation
5 Project Schedule

Table 5.1 covers the project schedule. The detailed project plan and project web page are completed. The image matching part and the basic classification is completed. An online training part will be further developed.

Table 5.1. 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</td>
<td>Implementation of the image matching machine learning model</td>
<td>Completed</td>
</tr>
<tr>
<td>30 Dec 2022</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</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>the record of received items</td>
<td></td>
</tr>
<tr>
<td>9 Jan 2023 to</td>
<td>Completion of preliminary implementation and detailed interim report</td>
<td>Completed</td>
</tr>
<tr>
<td>22 Jan 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23 Jan 2023 to</td>
<td>Preparation on the first presentation</td>
<td>In Progress</td>
</tr>
<tr>
<td>31 Jan 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Feb 2023</td>
<td>First presentation</td>
<td>In Progress</td>
</tr>
<tr>
<td>23 Jan 2023 to</td>
<td>Application for users to submit and tracking the application status</td>
<td>In Progress</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></td>
</tr>
<tr>
<td>10 Apr 2023 to</td>
<td>Completion of finalized tested implementation and Final Report</td>
<td></td>
</tr>
<tr>
<td>18 Apr 2023</td>
<td>Preparation of Final Presentation</td>
<td></td>
</tr>
<tr>
<td>3 May 2023</td>
<td>Project Exhibition</td>
<td></td>
</tr>
</tbody>
</table>
6 Future Plan

6.1 Online learning for image classification model

To ensure our model be able to learn autonomously and adopt to fast-changing data, our group plans to adopt online continual learning to keep updating the predictor for the data input after deployment. To avoid class imbalance, a new type will be added only when at least 10 lost items with that type are found.

We have designed the methodology for applying the model with online learning by updating the model on a based on fixed-time interval (Fig. 6.1). We will input the photos added between the time interval to the top-layer of the model for fine-tuning. In case of there is a new category of lost items added, append a new dimensionality of the matrix of the top layer of the neural network.

![Fig. 6.1. Implementation methodology of online learning](image)

We expect the newly added category will have a limited number of photos and induce an issue of class imbalance. We will resolve this issue by upsampling the photo of new categories by Synthetic Minority Over-Sampling Technique (SMOTE) [5].
6.2 Application

We will finish the implementation of the font-end pages, back-end server as quickly as possible. Afterwards, we will follow the new modified design mentioned in section 3.1 to integrate the model with the app. After finalizing the model and integration, we will test and debug the application soon.

References


