Automated Mobile UI Testing of Cross-Platform Based on Computer Vision

-COMP8755 Individual Project

- Widgets and image recognizing driving automated mobile testing of cross-platform based on Yolov3

Name: Weinan Zhang
Student ID: U5687862
Supervisor: Associate Pro. Zhencheng Xing
Outline

- Introduction
- Methodology
- Evaluation
- Results
- Conclusion
- Future works
- References
Background

Mobile User Interface Testing

a) Manual testing means having a human tester running a series of operations and effectively manually verifying whether the app is acting in the right way and graphical displays are consistent with the requirement.

b) Computer vision methodology can also be extended to GUI testing for testers to simplify their tasks.
The objective of this project is to apply computer vision method to software engineering, which using object detection method to implement the automated testing on cross platforms.

- Dataset collection
- Dataset pre-processing
- Computer version method choosing
- Train the dataset
- Test the dataset
- UI widgets detected
- Similarity comparison
- Results analysis

Figure 2 UI widgets
Methodology

- **Training Phase**

  - Training Dataset
  - Training Dataset Pre-processing
  - YOLO V3 Model Implementation
  - Testing

  ![Figure 3 Training Phase](image)

- **Testing Phase**

  - Testing Dataset Collected
  - Testing the Same Platform UI Components
  - Testing the Cross Platform UI Components
  - SIFT Similarity Comparison
  - Evaluation

  ![Figure 4 Testing Phase](image)
In the pre-process of the dataset, assuming the upper left corner and lower right corner label of the bounding box are \((x_1, y_1), (x_2, y_2)\). Therefore, the coordinate of x after normalization will be:

- The center point of x coordinate after normalization: \(((x_2 + x_1)/2.0)/w\)
- The center point of y coordinate after normalization: \(((y_2 + y_1)/2.0)/h\)
- The target bounding width after normalization: \((x_2 - x_1)/w\)
- The target bounding height after normalization: \((y_2 - y_1)/h\)
YOLOV3 Model

a. Bounding box prediction

Yolov3 method predicts the bounding boxes by using dimension clusters as anchor boxes. It predicts 4 coordinates for each bounding box, $t_x, t_y, t_w, t_h$. The predictions will be done by the top left corner of the image $(C_x, C_y)$ and the width and height of the bounding box $p_w, p_h$. Yolov3 using logistic regression to predict the objectness score for every bounding box. When the bounding box overlaps the object more than other bounding box prior, the objectness score will be 1.
b. Architecture

Yolov3 does have 53 convolutional layers. In addition, Yolov3 uses the strategy which is similar with the multiple scales of SSD. It uses feature map which does have 3 scales (13*13, 26*26, 52*52).

Figure 7 YoloV3 Architecture[4]
Similarity Comparison (SIFT)

1. Keypoints selection

The basic solution is to find the local maxima and minima for the images. Then for the selection of the points, some of them are not robust enough to the noise, therefore, SIFT will eliminate the keypoints that have low contrast, or lie every close to the edge.

* Figure 8 Keypoints selection*

*: The source of the pictures are from ANU website
Similarity Comparison (SIFT)

2. Feature matching

In the automated software app testing, after the object detection. SIFT is expected to be used to evaluate the results. For example, the buttons are tested on different platforms. To ensure the button is detected correctly at the same app for different platforms, SIFT will be used for the feature matching to examine the results.

*: The source of the pictures are from ANU website
Evaluation and Results

a. Evaluation

1. Training dataset

2. Testing dataset

*: The source of the pictures are from Rico Dataset[5]
Evaluation

3. Classes defined

There are total 10 classes or widgets used in this project. The widgets are very common in most of the mobile UI dataset. Therefore, the widgets are selected to be used for the automated testing in this project.

<table>
<thead>
<tr>
<th>Widget/Class</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Button</td>
<td>Allow the user to return to the main UI.</td>
<td><img src="image" alt="Home Button" /></td>
</tr>
<tr>
<td>Click Button</td>
<td>Allows the users to forward the next page.</td>
<td><img src="image" alt="Next Page" /></td>
</tr>
<tr>
<td>Shopping Button</td>
<td>Allow the users to shopping by using this button.</td>
<td><img src="image" alt="Shopping Cart" /></td>
</tr>
<tr>
<td>Toggle Button</td>
<td>Toggles that does have two states by clicking them.</td>
<td><img src="image" alt="Toggle" /></td>
</tr>
<tr>
<td>Menu</td>
<td>The textual button across the bottom of top of UI</td>
<td><img src="image" alt="Menu" /></td>
</tr>
<tr>
<td>Tick Box</td>
<td>The button which the uses could select the states by clicking that.</td>
<td><img src="image" alt="Tick Box" /></td>
</tr>
<tr>
<td>Profile</td>
<td>The button which could lead the users to profile.</td>
<td><img src="image" alt="Profile" /></td>
</tr>
<tr>
<td>Return</td>
<td>The button that could help users to return to previous page.</td>
<td><img src="image" alt="Return" /></td>
</tr>
<tr>
<td>Search</td>
<td>The button that could search the key information on UI.</td>
<td><img src="image" alt="Search" /></td>
</tr>
<tr>
<td>Slider</td>
<td>The button that allows users to adjust the level.</td>
<td><img src="image" alt="Slider" /></td>
</tr>
</tbody>
</table>

Table 1 Defined classes
4. Stopping Criteria

To avoid overtraining, stop training early provide a clear solution to this issue. A large number of training iterations or epochs is used then usual when training the network.

![Figure 10 Loss curves of 0-1000 epochs](image1)
![Figure 11 Loss curves of 100-300 epochs](image2)

<table>
<thead>
<tr>
<th>Region Avg</th>
<th>Class</th>
<th>Obj</th>
<th>No</th>
<th>Avg Recall</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.363987</td>
<td>0.898221</td>
<td>0.890010</td>
<td>0.006567</td>
<td>1.000000</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2 The varying indicators of errors
Evaluation

5. Evaluation metrics

\[
mAP = \frac{\sum_{q=1}^{Q} \text{AveP}(q)}{Q} \quad [6]
\]

mAP (mean average precision) is commonly used in measuring the accuracy of object detectors. mAP calculates the mean of the average precision value. For this project, mAP is the mean value of average precisions for each class (widget defined before). The mAP is defined as below, where \( Q \) is the number of queries in the set and \( \text{AveP}(q) \) is the average precision (AP) for a given query \( q \).

Figure 12  Evaluation metrics*

*: The pictures are made based on the theory of IOU, Recall, and Precision[6]
Evaluation and Results

b. Results.

1. Android Platform

The testing UI dataset are chosen from Rico dataset which are not used during the training process. This is only for the initial validation for this model.

<table>
<thead>
<tr>
<th>mAP threshold</th>
<th>mAP</th>
<th>Average IoU (%)</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>0.989899</td>
<td>75.60</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>0.75</td>
<td>0.978523</td>
<td>75.67</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>0.95</td>
<td>0.987597</td>
<td>76.85</td>
<td>1.00</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Figure 13  Testing on Android platform

Table 3 Evaluation metrics results for Android platform
Results

2. IOS APP Platform

The IOS app UI dataset are collected from the screenshots of Amazon Apps. The testing dataset is to simulate the users’ actions trace. During the trace of each action, the UI dataset has been recorded. This will more valuable for the automated testing of UI.

![Image of testing on IOS APP platform]

<table>
<thead>
<tr>
<th>mAP Threshold</th>
<th>mAP</th>
<th>Average IoU (%)</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0.50</strong></td>
<td>0.789899</td>
<td>66.15</td>
<td>0.85</td>
<td>0.76</td>
</tr>
<tr>
<td><strong>0.75</strong></td>
<td>0.786898</td>
<td>68.94</td>
<td>0.89</td>
<td>0.74</td>
</tr>
<tr>
<td><strong>0.95</strong></td>
<td>0.773564</td>
<td>72.70</td>
<td>0.95</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Table 4 Evaluation metrics results for IOS APP platform
Results

3. IOS Web Platform

The testing dataset for IOS web platform are collected from the same app with IOS app platform. This is aimed to evaluate the detected results for same app in different platform. Furthermore, the testing dataset for IOS web platform also simulate the trace of the users.

Table 4 Evaluation metrics results for IOS WEB platform

<table>
<thead>
<tr>
<th>mAP Threshold</th>
<th>mAP</th>
<th>Average IoU (%)</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>0.657256</td>
<td>60.56</td>
<td>0.77</td>
<td>0.67</td>
</tr>
<tr>
<td>0.75</td>
<td>0.662314</td>
<td>61.25</td>
<td>0.81</td>
<td>0.64</td>
</tr>
<tr>
<td>0.95</td>
<td>0.642531</td>
<td>65.88</td>
<td>0.84</td>
<td>0.62</td>
</tr>
</tbody>
</table>

Figure 15 Testing on IOS WEB platform
Results

4. Similarity Comparison Results

Using SIFT method to compare the similarity of same class of widgets on different platform. It is found that the similarity is relatively high although they their appearance have some difference. This model still could detect the widget successfully.
Conclusion

- First, using the UI dataset from Android platform to test this model. Because the model is trained by using the dataset from Android platform, the performance is the best of all the dataset, achieving the highest accuracy ($mAP=0.98$).

- Second, the model is tested by the dataset from IOS App platform. Although the performance is worse than Android, almost beyond 80% of widgets are detected successfully. This proves this method also could be used on different platform, although some widgets do have different appearances ($mAP=0.78$).

- Third, the model is being tested by the dataset from IOS web platform, detecting around 70% widgets. The performance is worst but most of the widgets could be detected ($mAP=0.65$).

- Finally, the Similarity comparison also be used to evaluate the feature matching of these widgets. It shows that the widgets that detected in different platform are relatively matched.
Future work

1) More widgets will be added in the pre-defined classes, there are many widgets in the actual mobile user interface.

2) Test this approach on platforms like Windows Phone, BlackBerry and Symbian.

3) Using the UI dataset from different platforms to train model to improve prediction performance.

4) In future, the YOLOV3 structure needs to be optimised to improve predictive efficiency.

5) The text input area detected will be added in future, which will use EAST algorithm.
References


