**Standard Operating Procedure on Image Classification Algorithm**

This document serves as the standard operating procedure, to identify the class of a bag

picture, based on an image classification model trained from bag images with class label A, B

or C. In our demonstration, 74 bags are trained in the model.

The procedure is divided into 2 parts: Image Processing and Image Classification.

**1. Image Processing**

**1.1 Methodology**

In the given set of images, each image contains more than 1 bags. For example, photo ‘CK2

20150721’ contains 4 A-graded shoulder bags of different colours (black/ blue/ clay/ blush).



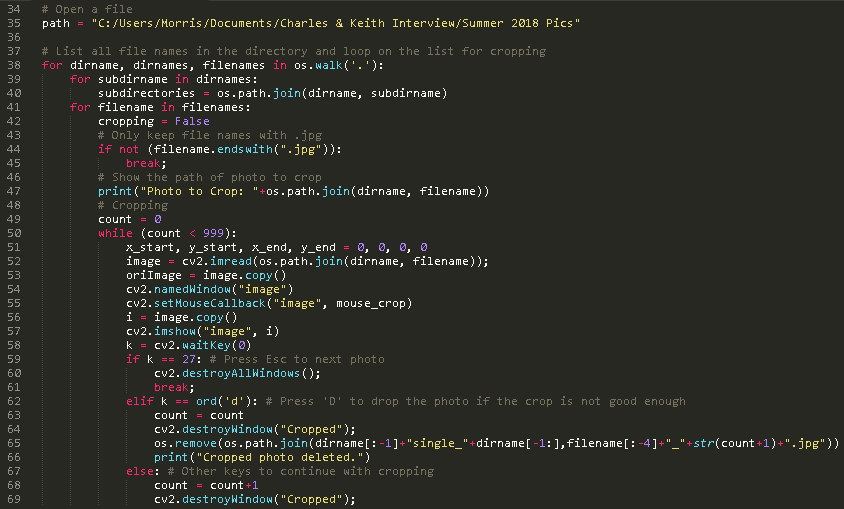
Firstly, we have to obtain individual image of each bag.

Crop\_Image.py, an image cropping python function is developed on python, which enables

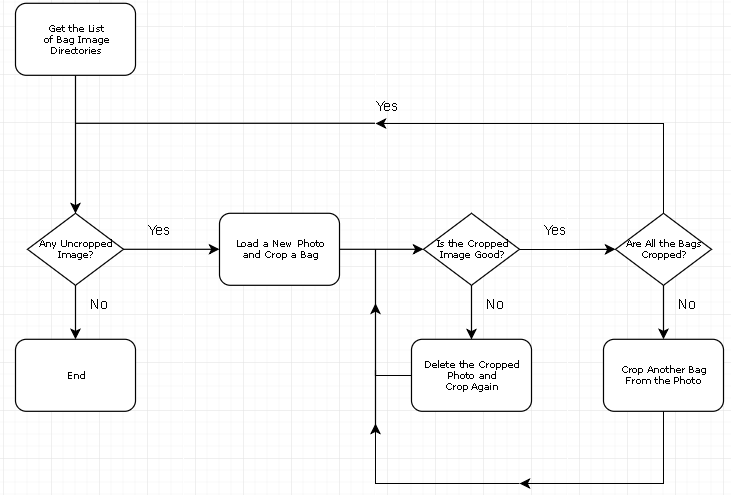
us to drag and crop the image of interest, and save it as an individual image for further

usage.

In the function, libraries like os, sys, opencv and numpy are imported.



The function flowchart of Crop\_Image.py is shown below:



**1.2 Procedure**

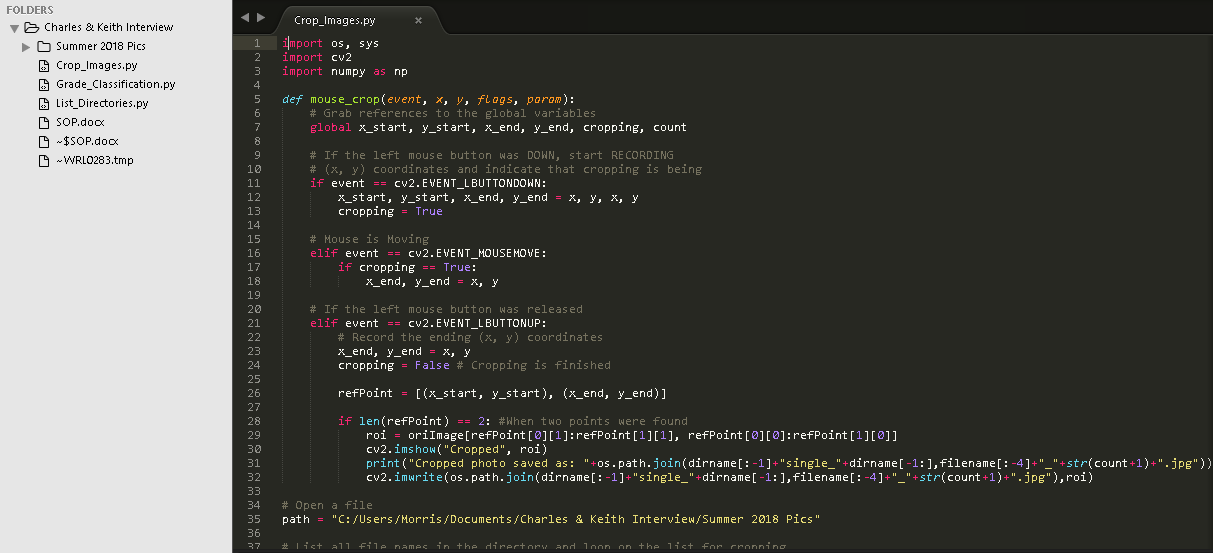
Before carrying out the procedure, it is required to create folders ‘Single\_A’, ‘Single\_B’ and

‘Single\_C’ in the directory, for storing the cropped images from the folders ‘A’, ‘B’ and ‘C’.

The procedure of using the ‘Crop\_Images’ function is shown below, under the environment

of Python 3.5:

1. Load the function file on Sublime Text, and build the code (Ctrl+B).



1. A photo is loaded. Drag an individual bag from top left to bottom right, and crop.



1. Check if the quality of the cropped image is good. Press ‘D’ to crop again if the

quality is not good enough, and press any key other than Esc to continue cropping.



1. Press Esc to go to a new photo, after all the bags are cropped.
2. Repeat step 2 to 4, until all the original photos are cropped. Close the image pop-up.

By doing so, we successfully obtain 34 images of class A, 24 of class B and 47 of class C, in

the folders ‘Single\_A’, ‘Single\_B’ and ‘Single\_C’ respectively.

In the demonstration, the total number of individual bags is 105. The proportion among the

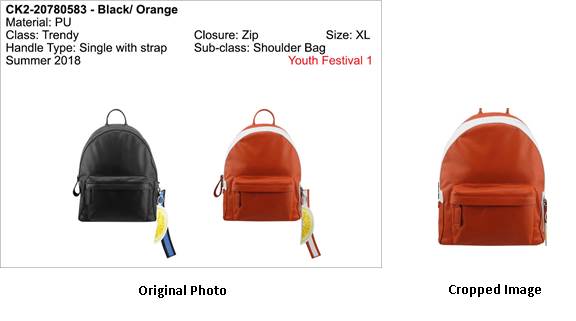
classes is slightly imbalanced (32%/23%/45%), yet it should not affect the accuracy much.

**1.3 Notes on Cropping**

It is later found that, cropped images with handle often resulted in lower test accuracy in

classifying the class than images with little/no handle. As a result, the images are cropped

with as little handle as possible. Here are some examples:





**2. Image Classification**

**2.1 Data Stratification**

Before classifying the class of the images, we divide the images into 3 groups: training data,

validation and testing data, which account for 70%, 10% and 20% of data respectively.

Training samples are for training the classification model, validation samples can help

running a holdout on the training set, and test samples evaluate the performance of the

model.

Table below shows the division of the data among the groups:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | No. of Images | Train (70%) | Validation (10%) | Test (20%) |
| A | 34 | 24 | 3 | 7 |
| B | 24 | 17 | 2 | 5 |
| C | 47 | 33 | 5 | 9 |
| Total | 105 | 74 | 10 | 21 |

We then create folders ‘training\_set’, ‘validation\_set’ and ‘test\_set’, and subsidiary folders

‘A’, ‘B’ and ‘C’ in each of them. After that, we allocate the images from folders ‘Single\_A’,

‘Single\_B’, and ‘Single\_C’ according to the table above.

**2.2 Building Classification Model**

Build the code in the Class\_Classification.py file, and a model called ‘classifier.h5’ will be

created in the directory. The classification result on test data can also be viewed on Sublime

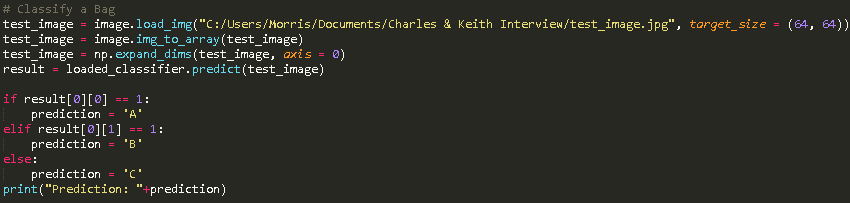
Text.

**2.2 To Classify a Bag**

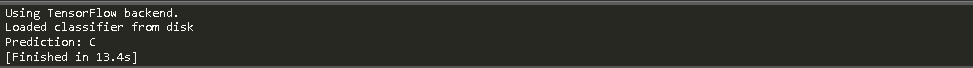
If we have a bag photo and want to classify its class, we can change the image directory in

line 25 of the Classify\_A\_Bag.py file and build the code on Sublime Text. The classification

model built in the previous step will be loaded.



Build the code and the class prediction is generated.



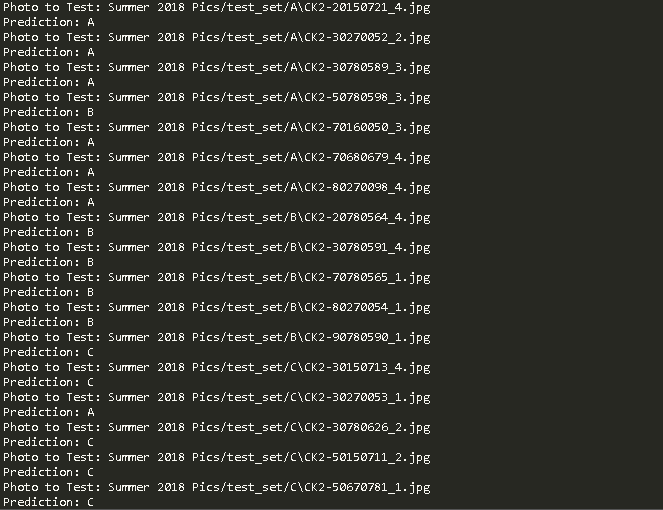
**3. Classification Model Evaluation**

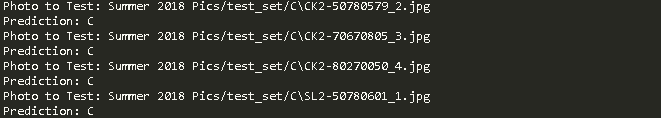
We set the following parameters in the image classification function, in the

Class\_Classification.py file.

* training set batch size = 32
* validation set batch size = 10 (the total number of images in the validation set
* steps for epoch = 74 / 32 ≈ 3 (rounding up)
* epochs = 200
* validation steps = 10 / 10 = 1

Then we build the code on Sublime Text. Here is the test result:





Out of 21 test images, 18 are correctly recognized and the test accuracy is 18 / 21 ≈ 86%