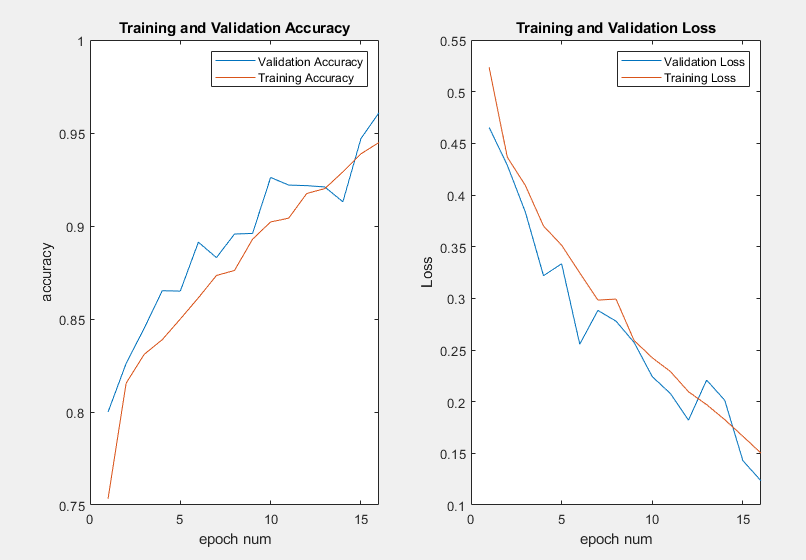
Training:

As table 1 and figure 1 below show, when running 16 epoch using Keras Conv2D, we can reach the accuracy of validation as high as 0.96, which is applicable for a daily-life classification program. However, if the program is going to be applied for classification factorials, higher accuracy is required. The trend of increasing accuracy and decreasing loss show no sign of reaching limitation. To improve the accuracy, simply run more epochs. Luckily, for the training data of thousands level, the run time for one epoch is acceptable. Also, in order to be applicable for wider range of categories, and more image conditions like not well focused, or more noisy images, larger dataset with those problematic images can be added to the training data, so the program can be more applicable in various environment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Epoch num. | Runtime(sec) | Training accuracy | Validation accuracy | Training loss | Validation loss |
| 1 | 40 | 0.7534 | 0.8000 | 0.5235 | 0.4654 |
| 2 | 38 | 0.8155 | 0.8261 | 0.4367 | 0.4289 |
| 3 | 40 | 0.8310 | 0.8450 | 0.4092 | 0.3834 |
| 4 | 41 | 0.8389 | 0.8652 | 0.3700 | 0.3219 |
| 5 | 42 | 0.8500 | 0.8650 | 0.3514 | 0.3335 |
| 6 | 41 | 0.8614 | 0.8913 | 0.3246 | 0.2556 |
| 7 | 41 | 0.8734 | 0.8830 | 0.2983 | 0.2883 |
| 8 | 39 | 0.8761 | 0.8957 | 0.2993 | 0.2779 |
| 9 | 39 | 0.8929 | 0.8960 | 0.2591 | 0.2573 |
| 10 | 37 | 0.9022 | 0.9261 | 0.2425 | 0.2242 |
| 11 | 38 | 0.9042 | 0.9220 | 0.2292 | 0.2079 |
| 12 | 36 | 0.9175 | 0.9217 | 0.2096 | 0.1821 |
| 13 | 35 | 0.9201 | 0.9210 | 0.1971 | 0.2209 |
| 14 | 37 | 0.9291 | 0.9130 | 0.1826 | 0.2013 |
| 15 | 38 | 0.9388 | 0.9470 | 0.1665 | 0.1429 |
| 16 | 37 | 0.9449 | 0.9609 | 0.1502 | 0.1234 |

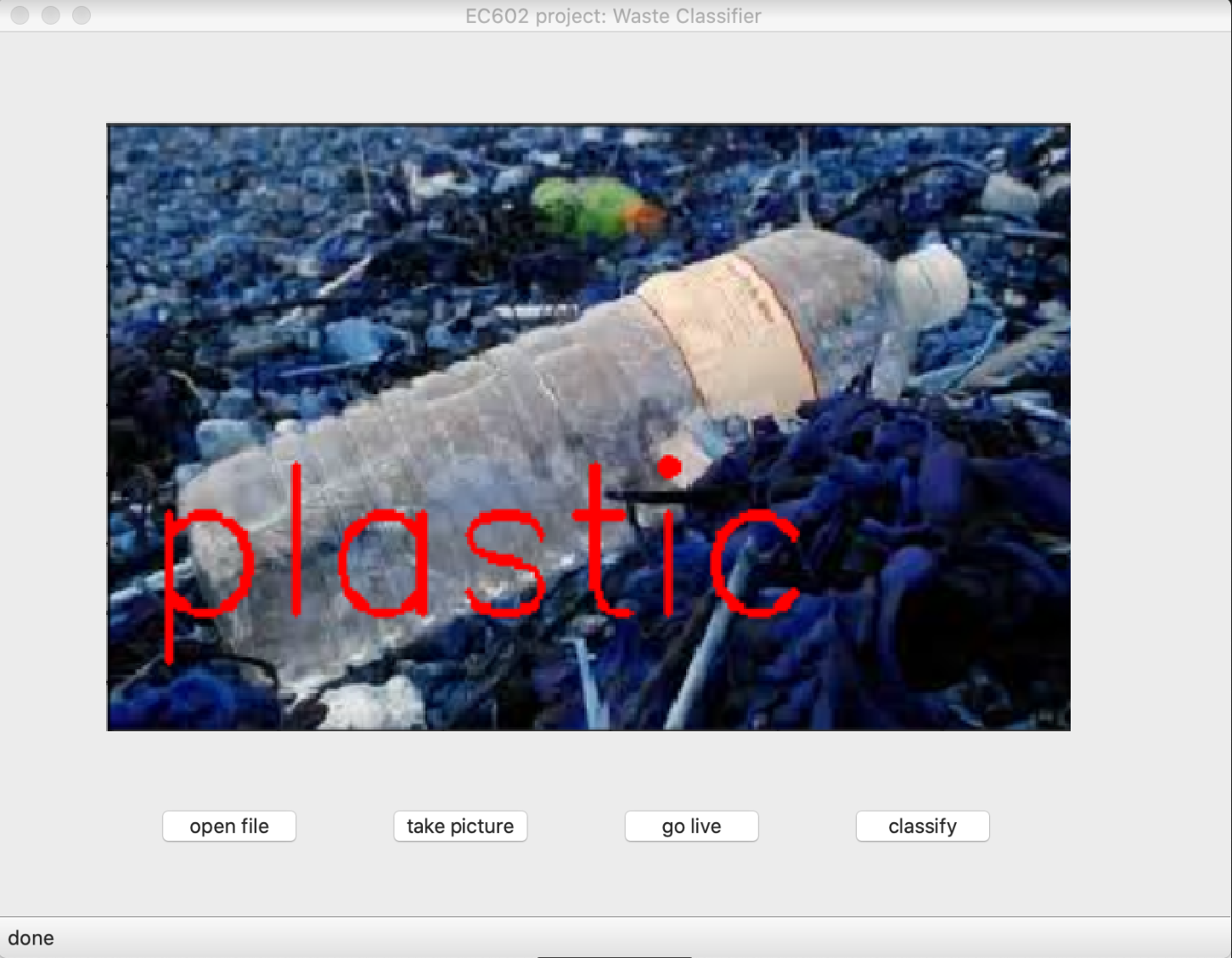
**Table 1,** Result of running 16 epoch.



**Figure 1**, Accuracy and loss for training and validation process.

Prediction

The prediction process for a single photo takes ignorable time, which also shows the success of the program. The photo taken, even with noisy background as the figure shows below, the model is still able to predict the category of the waste correctly. The figure below is an example of predicting a image of plastic bottle which is photographed in a noisy background. The program take a photo either from local file, or from camera, and when the *classify* button is pressed; the image will be labeled by the program as follow. Unfortunately we do not able to implement multiple object identification and we did not have trained for non-waste category, so if multiple wastes are in the image, or no waste in the image, the program cannot identify those situations and will label the image to the most likely category according to the model, which will be an incorrect result.



**Figure 2,** Sample output of the program.

User interface:

The interface is implemented by pyqt5. The interface enables 3 ways of input: open local file, take a picture and use live streaming of computer camera. *Classify* button should be pressed if user want to have the loaded image be classified and labeled. The interface is shown in figure 2. The live function is still in progress, since we do not have a good algorithm detecting multiple objects and no waste, the real-time classification will produce a lot of incorrect result. The ideal interface will not have *classify* button but label the image once it is loaded. And the program should be able to run in smartphones and tablets. Those functions can be implemented if time get prolonged.