EGuard Documentation on Technologies Used

1 Overview

Mailu is used in EGuard's prototype as the mail server since Mailu is open source, easy-to-install and full-featured. A Linux machine is used to host the mail server since Linux is open source, flexible, secure and reliable for running servers.
Live demo: https://cnn-repo.s3.ap-east-1.amazonaws.com/build/index.html

2 Architecture

2.1 Mailu

Mailu version 1.7, which is an open source mail server as a set of Docker images, is installed in a Linux machine. Mailu uses postfix as the mail transfer agent (MTA) and is configured to use roundcube as the webmail. Mailu contains other microservices such as a proxy server and a web administration interface and has security features such as enforced TLS and anti-spoofing.

2.2 Mail Transfer Agent

Using postfix as the MTA is not a must. Choice of the MTA does not matter much to the scam detection utility being developed. The MTA can be replaced by other MTAs such as exim and sendmail if Mailu is not used as clients' mail server.

2.3 Webmail

A webmail is a web-based service that allows users to access their emails. Interfaces have to be developed for each choice of the webmail. In EGuard's prototype, roundcube is used. Roundcube plugins are written in php and javascript and installed by adding a folder inside the roundcube plugin directory. A warning banner is developed to display a warning message at the beginning of the email content whenever an email sender is not recognized before. Menu buttons are developed to recognize or unrecognize email senders of selected emails. A recognized email sender list is persistently stored inside the linux machine which hosts the mail server. gRPC is used as the protocol for webmail to communicate with the linux machine to query or update the recognized email sender list. Whether a warning banner displays or not depends on the query result. It is possible for the recognized email sender list to be stored in any linux machine.

3 Implementations

3.1 Image Classification

In this project, an image classification example is already developed and an object detection example will be developed. For image classification example, 99% validation accuracy and 98% testing accuracy are already achieved and prediction of each image is completed within 0.07 second using tensorflow model in python and a desktop with i7-10700 CPU and 32GB RAM. Using tensorflow model in javascript and the same desktop, prediction time is approximately doubled. Since image classification tasks are generally more analytical rather than about real-time monitoring, such latency and testing time are sufficient. It is believed that 98% testing accuracy is likely sufficient for readers to persuade themselves, their bosses, their colleagues and even investors to kick-start their project, while keeping this example simple enough for readers to get started. Testing accuracy can be further improved by larger dataset, better hyper-parameter tuning and re-training more layers.

ResNet50 is used as the base model for re-training. ResNet50 has higher accuracy than AlexNet, VGGNet and GoogLeNet. Besides, it is shown that ResNet is relatively easy to train, tolerant of hyper-parameters and generalizes well [5]. ResNet50 is chosen among ResNet variants since it is believed that ResNet50 strikes a good balance between accuracy and training time. Deeper ResNet architectures have higher accuracy but longer training time. For start-up companies and individual learners with limited computational resources, training time and time due to trial and error should be restricted. In fine-tuning, the last fully connected dense layer for prediction is first removed. Then, a pooling layer, flatten layer, dense layer of 256 neurons, dropout layer and dense layer of 2 neurons for prediction are added in order. Only these added layers are trained while layers from the base model are frozen. Pooling layer reduces computational cost and reduces overfitting. Flatten layer reshapes the tensor from the pooling layer. Dropout layer reduces inter-dependent learning among the neurons and thus reduces overfitting. The first dense layer takes previous knowledge and provides learning features. The second dense layer classifies the image and outputs class labels.

3.2 Object Detection

On the other hand, object detection tasks generally emphasize on latency and frames per second (FPS) to accomplish real-time monitoring. This example has not been implemented yet. The model is aimed to achieve 30 FPS and 10% mean average precision (mAP) measured at 50% intersection over union (IOU) using the same desktop with i7-10700 CPU and 32GB RAM [6]. Since it is expensive for startup companies and individual learners to set up high-end GPUs, it is difficult to achieve both high FPS and high mAP at the same time. It is believed that for a deployable solution, low mAP can still deal with typical cases well and can be compensated by other parts of the application, while a fairly good FPS should not be compromised. Otherwise, real-time user interactions and real-time control systems become infeasible as the whole pipeline is bottlenecked by this object detection task. It is hoped that readers succeed to develop real-time applications such as games and autonomous robot plugins.

YOLOv3-tiny will be used as the base model for re-training. Although YOLO cannot localize some objects precisely and the number of nearby objects that YOLO can predict is restricted [7], YOLO is much faster than other state-of-the-art systems. For example, YOLO is three times faster than SSD [8]. Systems such as R-CNN, Fast R-CNN and Faster R-CNN generate region proposals and then classify the objects inside. Despite the fact that these systems allow more fine-grained object detection, they are especially not considered in real-time systems mainly because generating region proposals are very time-consuming. YOLO is proved to accomplish real-time performance when connected to a webcam by the research team [7], which is promising. YOLOv3-tiny is chosen among YOLO variants to ensure that the model is sufficiently lightweight and fast to re-train.

4 References

[1] https://ijcsit.com/docs/Volume%207/vol7issue5/ijcsit20160705014.pdf
[2] https://openreview.net/pdf?id=ryxyCeHtPB
[3] https://www.researchgate.net/profile/Jordan-Bird/publication/ 325803364_A_Study_on_CNN_Transfer_Learning_fo
Study-on-CNN-Transfer-Learning-for-Image-Classification.pdf
[4] https://arxiv.org/pdf/1406.2952.pdf
[5] https://arxiv.org/pdf/1603.05027.pdf
[6] https://arxiv.org/pdf/1907.11093.pdf
[7] https://openaccess.thecvf.com/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf
[8] https://arxiv.org/pdf/1804.02767.pdf