Oregon State University
College of Engineering

CS Capstone Technology Review

December 4, 2018

Pedestrian Counting and Privacy Preservation

Prepared for

Oregon State University

Dr. Fuxin Li

Prepared by

Group 9

Pavement Prometheus

Mazen Alotaibi - Deep Learning Specialist

Abstract

This document outlines the technolgoies investigated and chosen for the execution of Pedestrian Counting and Privacy Preservation project. The document will discuss about the selection of programming language, deep learning framework, and real-time object detection algorithm. In addition, the document will compare and contrast multiple options within each category based on certain criteria.

CONTENTS

1 INTRODUCTION

The software described in this document, Facial Detector, and Obfuscator, is a project under the advisement of Chanho Kim (Georgia Tech) and Dr. Fuxin Li (Oregon State University). The client for this project is the City of Portland, which wants a proof of concept for a way to transform the data from their traffic cameras so the city may store the data without storing identifying information about the citizens in the footage.

1.1 System Purpose

Our team will design a pedestrian/vehicle detection model which is able to obfuscate all identifying features of pedestrians and vehicles for a given video feed. This will allow for storage of the video data without storing identifying information on the pedestrians.

1.2 System Scope

The scope for this project is immediately to have a system that results in information on pedestrian movements that can be stored for open access by the public. An update that is not necessary, but is desirable, is the ability to provide data on traffic as well.

1.3 Work Log

My role in this project is to setup the environment of development of the simulation tool, data collection, and supervise over my teammates on the development of pedestrian detection algorithm and face detection algorithm.

Project Section	**Maintainer**
Simluation Setup	Mazen Alotaibi
Data Collection	All team members
Pedestrian Detection Algorithm	Mazen Alotaibi and Miles Davies
Face Detection Algorithm	Mazen Alotaibi and Stephanie Allison Hughes
Obfuscator Algorithm	Ian McQuoid
Source Control and Software Design	Ian McQuoid

2 REFERENCES

[1] C. Voskoglou, "What is the best programming language for machine learning?" Available at https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7 (2017/05/05).

[2] P. Kanada, "Which programming language is considered to be best for machine learning?" Available at https://www.datasciencecentral.com/profiles/blogs/which-programming-language-is-considered-to-be-best-for-machine (2018/07/28).

[3] BizDevCorp, "The 5 best programming languages for ai development," Available at https://www.futureproofing.io/blog/the-5-best-programming-languages-for-ai-development (2018/06/05).

[4] "Top 8 deep learning frameworks," Available at https://www.marutitech.com/top-8-deep-learning-frameworks/.

[5] M. Opala, "Deep learning frameworks comparison tensorflow, pytorch, keras, mxnet, the microsoft cognitive toolkit, caffe, deeplearning4j, chainer," Available at https://www.netguru.co/blog/deep-learning-frameworks-comparison (2018/09/02).

[6] A. Ahmed, "Choosing a machine learning framework in 2018," Available at https://agi.io/2018/02/09/survey-machine-learning-frameworks/ (2018/02/09).

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *ArXiv e-prints*, Jun. 2015.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *ArXiv e-prints*, Mar. 2017.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," *ArXiv e-prints*, Dec. 2015.

[10] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv e-prints*, Apr. 2018.

[11] J. Hui, "What do we learn from single shot object detectors (ssd, yolov3), fpn focal loss (retinanet)?" Available at https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d (2018/03/28).

[12] ——, "Object detection: speed and accuracy comparison (faster r-cnn, r-fcn, ssd, fpn, retinanet and yolov3)," Available at https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 (2018/06/19).

[13] ——, "What do we learn from region based object detectors (faster r-cnn, r-fcn, fpn)?" Available at https://medium.com/@jonathan_hui/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9 (2018/03/28).

[14] F. Li, Group meeting, October 2018.

3 Programming Language

3.1 Options

We have four options for the selection of the programming language that will be used, C++, Java, R, MATLAB, and Python. According to a survey of 2,000+ data scientists and machine learning developers about which languages they use and what project they are working on [1], 58% use Python, 44% use C/C++, 33% use R, and 13% use Java in general. When asked about prioritization for development, 33% prioritize Python, 19% prioritize C/C++, 5% prioritize R, and less than 1% prioritize Java. Moreover, for general usage, Python is usually used because it is an easier and faster way to build highly-performing algorithms because of its libraries support. C/C++ is mostly used in AI in games, robot locomotion, and embedded computing because of its level of control, high performance, and efficiency. R is usually used in bioengineering and bioinformatics because of its long history with biomedical statistics. Java is usually used in network security, fraud detection, and enterprise focus projects.

3.2 Criteria Being Evaluated

We will evaluate each option based on speed, learning curve, costing, community support, production ready, and Deep Neural Networks (DNN) frameworks support.

	C++	**Java**	**R**	**MATLAB**	**Python**
Speed	1	0	1	1	0
Learning Curve	0	1	1	1	1
Costing	1	1	1	0	1
Community Support	1	0	1	1	1
Production Ready	1	1	0	0	1
DNN Frameworks Support	1	0	0	0	1
Total	**5**	**3**	**4**	**3**	**5**

TABLE 1: Comparison between Programming Languages based on selected criteria

3.3 Discussion

C++ is the most powerful programming language out of these options because it is the only low-level programming language selected, which means high performance and efficiency. C++ is supported by the community and most of Deep Neural Networks frameworks, such as PyTorch and Caffe. However, C++ has a sharp learning curve, which

makes the development time slower. Java is easy to learn to learn and production ready, but Java isn't supported by any of the Deep Neural Networks frameworks. R is designed for statistical analysis and visualizations, but R isn't supported by any of the Deep Neural Networks frameworks and it isn't ready for production. MATLAB is designed for mathematical modeling and computer vision. MATLAB is supported by the community and Caffe, a Deep Neural Networks framework. However, MATLAB is the only language that you need to pay for its license. Python is the only programming language from this list that has the largest community support and Deep Neural Networks framework support. In addition, Python is easy to implement and production ready. However, Python is slow compared to C++ [2][3].

From my research, the best two programming languages based on our criteria are C++ and Python. C++ is fast but hard to develop. Python has every criterion checked besides speed. However, our client, City of Portland, wants a proof of concept of a way to transform the data from their traffic cameras so the city may store the data with storing identifying information about the citizens in the footage. Therefore, we will be using Python as our best option.

4 Deep Learning Framework

4.1 Options

We have four options for the selection of the deep learning framework that will be used, TensorFlow, Keras, and PyTorch. TensorFlow, Keras, and PyTorch created by Google in November 2015, a Google engineer (Francois Chollet) in December 2017, and Facebook in July 2018 respectively. TensorFlow and PyTorch are adopted by several giant companies, such as Twitter, IBM, AirBus because of their highly flexible system architecture [4]. TensorFlow is mostly used in voice/image recognition, text classification/summarizing, and natural language processing because of the development tools supported by Google Translate. Keras is used for prototyping because its high-level syntax and architecture, however Keras isn't always easy to customize the Neural Network because of its high-level architecture. Therefore, researchers tend to use Keras as front-end and TensorFlow as back-end for fast development and easy to customize the Neural Networks. In other hand, PyTorch already has these features as it is used for training deep learning models quickly and effectively so it is the framework of choice for a large number of researchers. In addition, TensorFlow, Keras, and PyTorch are fully support Python only [5].

4.2 Criteria Being Evaluated

We will evaluate each option based on speed, learning curve, documentation, community support, framework logic structure, and full customization of Neural Networks (NN).

	Keras	**TensorFlow**	**Keras/TensorFlow**	**PyTorch**
Speed	0	1	1	1
Learning Curve	1	0	1	1
Documentation	1	1	1	0
Community Support	1	1	1	1
Framework Logic Structure	0	0	0	1
Full Customization of NN	0	0	0	1
Total	**4**	**3**	**5**	**5**

TABLE 2: Comparison between Deep Learning Frameworks based on selected criteria

4.3 Discussion

Keras is the easiest to learn and implement compared to TensorFlow and PyTorch and it is supported by the community. However, Keras isn't fast and doesn't allow full customization of Neural Networks. In the other hand, TensorFlow is fast and has the largest community support. However, TensorFlow is hard to implement because TensorFlow's calls structure and it doesn't allow full customization of Neural Networks. Moreover, Keras and TensorFlow can be used together, which will allow us to use Keras easy syntax and TensorFlow speed. But we can't fully customize the Neural Networks. PyTorch has all TensorFlow and Keras features besides documentation and. Although PyTorch is created by Facebook AI Research Lab, it is still young, 2018, compared to TensorFlow, 2015. Nonetheless, PyTorch is the only framework that allows full customization of Neural Networks and has the best logical structure of the framework of the data pipeline [4] [5] [6].

From my research, the best two framework structure based on our criteria are Keras/TensorFlow and PyTorch. Keras/TensorFlow has simple syntax and easy to implement, and PyTorch has the best logical structure of the framework and the only that allow full customization of the Neural Networks. Although Keras/TensorFlow will help us start the project fast as our advisor, Chanho Kim, uses TensorFlow mainly, selecting PyTorch will be the optimal option because PyTorch allows us to fully customize the Neural Networks, which we might need to when we change our current computer vision system for research purposes.

5 Real-time Object Detection Algorithm

5.1 Options

We have four options for the selection of the real-time object detection algorithm that will be used, Faster Regional-Convolution Neural Networks (Faster R-CNN)[7], Mask R-CNN[8], Single Shot MultiBox Detector (SSD)[9], and Yon Only Look Once (YOLOv3)[10]. Faster R-CNN uses Convolution Neural Networks as feature extractor and Region Proposal Network (RPN), a region proposal method by an internal Deep Network and the Region of Interest (RoIs) are derived from the feature maps instead. SSD uses VGG19 netowrk as feature extractor and convolution layers followed by convolution filters to make the prediction. YOLOv3 uses DarkNet 53 as backbone for feature extraction and Feature Pyramid Networks (FPN), composes of a bottom-up and a top-down pathway, to detect small object better [11].

According to a research comparison [12], Faster R-CNN has a small accuracy advantage if real-time speed isn't needed. SSD has a good frames per seconds for lower resolution images at cost of accuracy in real-time processing. However, SSD performs much worse on small object detection. For both algorithms, they both perform better when the input image resolution is high and SSD accuracy for detecting small object increases. In the other hand, when decreasing the resolution of the input image into half, the accuracy is decreased by 15.88% on average for both Faster R-CNN and SSD.

5.2 Criteria Being Evaluated

We will evaluate each option based on implementation, real-time detection of small, medium, and large objects, segmentation, frames, and mean average precision (mAP).

	Faster R-CNN	Mask R-CNN	SSD	YOLOv3
Implementation Complexity	1	0	0	0
Detection of Small Objects	1	1	0	0
Detection of medium Objects	1	1	1	1
Detection of Large Objects	1	1	1	1
Segmentation	0	1	0	0
Speed	0	0	1	1
Total	**4**	**4**	**3**	**3**
Frames	17	0	59	91
mAP	34.9	0	26.8	33

TABLE 3: Comparison between Real-time Object Detection Algorithms based on selected criteria

5.3 Discussion

Faster R-CNN [7] has the highest accuracy compared to SSD [9] and YOLOv3 [10], and Faster R-CNN is easy to implement and the only one that can detect small objects. However, Faster R-CNN has the lowest frames per second, which means a lot of spatial information of continue object detection lost due to this. In addition, SSD and YOLOv3 have similar results, but YOLOv3 have more frames per second and higher accuracy compared to SSD [12] [13] [11].

According to Dr. Li [14], our team is required to find a middle point allowing us to preserve the privacy of pedestrians while generating useful traffic data. We will build a computer vision model for detecting pedestrians within each frame. State-of-the-art computer vision models, such as You Only Look Once (YOLOv3) [10] and Single Shot MultiBox Detector (SSD) [9], have poor accuracy in detecting objects. These models average 33% at 91 frames per seconds and 26.8% at 59 frames per seconds, respectively, compared to other Computer Vision models that have a higher accuracy, but need more computation time for object detection. Therefore, pedestrian detection using YOLOv3 and SSD increases the chances of having more false negatives, not blurring pedestrians' faces, due to the low accuracy of real-time analysis. To reduce the chances of having false negatives generated by our model we will need to separate the object detection model into two stages: pedestrian detection and face detection.

From my research, I conclude that YOLOv3 is the optimal real-time object detection algorithm because the project is mostly focused on real-time object detection. In addition, by implementing YOLOv3 using PyTorch, we will be able to change the backbone of YOLOv3 with better backbone that suits our problem. Moreover, we will be detecting pedestrians within each frame, which means we will insure a higher accuracy than 33% as pedestrians are medium size objects.

6 Conclusion

To sum up, we have decided to select Python, PyTorch, and YOLOv3 as our main programming language, Deep Learning framework, and real-time object detection algorithm respectively. Python because it is easy to learn and develop, supported by the community, and supported by Deep Learning frameworks. PyTorch because it allows us to develop and fully customize the Neural Networks. Finally, YOLOv3 because it is the best real-time object detection algorithm from my research.