Rezka Bunaiya Prayudha

AddInsight-Vision

An Intelligent Video System for Object Detection in Traffic Scenes and Vehicle Re-Identification

Thesis submitted for the degree of Master of Engineering

Master of Engineering (Electronics)

College of Science and Engineering

Academic Supervisor: Dr. Nasser Asgari
Industrial Supervisor: Dr. Chong Liaw

2021

I, Rezka Bunaiya Prayudha, certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university and, to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

I want to thank Nasser, Chong, and all the staff from the College of Science and Engineering Flinders University and SAGE Automation for their tremendous help during this project development. I also want to thank the Australia Awards Scholarship for letting me pursue my master's program here at Flinders.

To Mom, Miranda, my family and colleague.. I'll spend the rest of my years trying to figure out how to thank you

Executive Summary

This thesis project involved the development of the AddInsight-Vision - an intelligent video system designed to extract information from traffic scenes which is part of SAGE Automation's Intelligent Transport System. The primary aim of the project was to design and develop such a stream-processing pipeline which functioned to detect traffic objects, including vehicle and pedestrian accurately and send the information to the management system for further analysis.

The initial stage of AddInsight-Vision development covered the state-of-art analysis and determining evaluation metrics—the state-of-art analysis including proposed hardware requirements, tools and software framework. This project used Average Precision (AP) and Cumulative Matching Characteristic (CMC) to evaluate AddInsight-Vision.

Following the development of AddInsight-Vision, the stream processing pipeline was constructed and tested. Early testing result explicitly showed that AddInsight-Vision were not able to detect object such as bus and pedestrian in certain conditions such as night times.

After initial testing, AddInsight-Vision was modified to improve object detection accuracy and overall performance. The object detection and vehicle re-identification were models re-trained using transfer learning based on DPTI and Ve-RI datasets.

In the final evaluation, the average precision of object detection model achieved over 90% for car and bus detection while maintaining high Frame Rate per Second (FPS). In addition, the CMC result of vehicle re-identification model achieved second-best in contrast with other baselines with a similar task. As such, the overall performance of AddInsight-Vision during testing and evaluation shows the project development was successful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, sensor technologies for Intelligent Transportation System (ITS) has developed massively and widely used in many applications such as traffic analysis, incident detection, and transportation model that will be useful for simulate policy and infrastructure investment (Ibrahim et al, 2020). McGowen and Sanderson (2011) explain that typically ITS are equipped with sensors such as cameras or pneumatic tubes to capture traffic information and a computing device for analyzing captured data.

SAGE Automation is one of the leading ITS companies in South Australia, utilizing the AddInsight system to provide traffic analytics information such as travel time and the number of the pedestrian. Sensor technologies used in the AddInsight system is an Edge device capable of capturing Bluetooth and WiFi data from the On-Board Unit (OBU) installed on a vehicle or carried by a pedestrian. The Edge device is placed near the roadside, making it able to obtain the data from passing cars or pedestrians. The captured data is then transmitted to the SAGE Automation analytics dashboard for further analysis. However, the current AddInsight system requires the user to have installed Bluetooth-based OBU device to be detected by the Edge device. Also, reference data is needed to be compared with the data from the Edge device.

In this project, the proposed solution to improve SAGE AddInsight is an Intelligent Video System (IVS). IVS creates an analytics system that uses the camera as the sensing device and computer vision to process the data. The camera is used to monitor the traffic scene and extract real-time information. The camera data then processed using computer vision to obtain a vehicle and other traffic object features and spatial data. By implementing IVS in Intelligent Traffic System, the traffic parameters such as density and traffic flow could be

estimated accurately.

1.1 Research Objectives and Contribution

This project aims to develop an end-to-end IVS pipeline that could perform specific tasks, i.e., capture video streams from the camera, process the data to obtain the traffic information using an object detection model, visualize and display the results, and gather the metrics. IVS pipeline is equipped with object detection models based on the Convolutional Neural Network to increase the accuracy of traffic-analytics. The object detection system is trained using transfer learning to enable the specific traffic data approach and reduce the detection error. The transfer learning process of object detection includes training using several pre-trained on the dataset based on the South Australia Department of Planning, Transport, and Infrastructure traffic scene videos. This project also investigates the vehicle re-identification system development by implementing an improved loss function to increase the metrics result. Furthermore, the following contributions were made during this thesis project development.

- Create an effective computer vision testing suite, which includes model evaluation and model testing.
- Create an end-to-end Intelligent Video System that can capture the traffic information and integrate with the AddInsight system and achieved well-performing Frame Per Second (FPS).
- Perform dataset curation.
- Identifying transfer learning and evaluation of object detection algorithms, namely, DetectNet, Faster RCNN, Retinanet, and Single Shot Detector (SSD).
- Detail performance analysis of the vehicle re-identification model by considering the implementation of the improved loss function.

1.2 Outline

The rest of the thesis is organised as follows:

Chapter 2 present background knowledge and related works related to the development of this project.

Chapter 3 describe the strategies to develop and implement the proposed intelligent video system.

Chapter 4 demonstrates the implementation and evaluation of the intelligent video system.

Chapter 5 concludes this thesis and suggest some of the future works for further development.

Chapter 2

Literature Review

The literature review in this chapter commences with a background of Intelligent Transportation System (ITS) and the supporting component of ITS such as intelligent video system and deep learning. This chapter then proceeds to evaluate related works on streaming pipeline architecture including object detection, tracking and re-identification, with a focus to gaining insights from traffic scenes.

2.1 Background Knowledge

This section covers background knowledge of Intelligent Traffic System (ITS) components, including the management of ITS, the sensor technologies used in the ITS, and the implementation of intelligent video support in the ITS.

2.1.1 Intelligent Transport System

Intelligent Transport Systems (ITS) are advanced networking, electronics, navigation, and information processing technology designed to enhance the performance of the current transport system. Singh and Gupta's (2015) study points out that several subsidiaries of Intelligent Transportation Systems are commonly implemented around the world to solve traffic and transport problems. The subsidiaries are Advanced Traveller Information System (ATIS), Advanced Traffic Management System (ATMs), Advanced Public Transportation Systems (APTS), and Emergency Management System (EMS). The Advanced Traveller Information System (ATIS) implements a wide variety of technology, such as mobile phones and the internet, or other traffic data sources to assist travelers and drivers regarding the best available routes and accessible travel modes. Ghavami's (2020) study proposes the shortest path analysis method to determine

the best way of the multimodal transportation network based on the Geospatial Information data. Another study related to ATIS implementation was proposed by Adeleke et al. (2019), where they developed a web-based ATIS to provides information on route guidance, city services in the Metropolis, Nigeria.

The second subsidiaries of ITS are ATMS, commonly used by law enforcement agencies to manage traffic by tracking traffic flow and making necessary decisions promptly, such as change measures of traffic signals to enhance traffic flow. The tracking procedure of traffic flow could be channeled by integrating several sensors to gather updated traffic information. Anand et al. (2020) proposed their research on using Light Detection and Ranging (LiDAR) sensors to provide a 3-dimensional representation of traffic scene with the aims to improve traffic flow and safety. The following subsidiaries of ITS are APTS and EMS, which are the utilisations of ITS to increase public transportation's operational efficiency, with the latter more focused on providing help in emergency conditions. Furthermore, from the above explanations, it is shown that the advances and integration in sensor technologies areas create more opportunities for developing an intelligent transportation system.

2.1.2 Sensor Technologies for Intelligent Transportation System

Guerrero-Ibañez et al. (2018) state that the performance of ITS depends heavily on the platform used to access, capture, and reliable process data from the traffic scenes. Sensor technologies are a critical component to collect the data in the ITS system. The data are then presented to the transport management system for further analysis and eventual actions. Barbagli et al. (2012) explain that there are two categories of sensing platforms used to capture the traffic data based on the location of installation, intrusive and non-intrusive. The invasive sensor is a type of sensors installed on pavement surfaces. It has been widely implemented and has high accuracy in capturing traffic data. However, the major drawbacks are that intrusive sensors have high operation and maintenance costs and may cause traffic disruption during the installation. Some examples

of invasive sensors are passive magnetic sensors, pneumatic tube sensors, and inductive coils loops. These sensors are connected either wired or wireless to the processing unit.

The second type of sensor technologies of ITS is non-intrusive sensors. A non-intrusive sensor is a type of sensor mounted at various positions on the roads other than above it. Some examples of non-intrusive sensors are camera, infrared, Radio-frequency identification, and ultrasonic sensor. This sensor could detect and classify vehicles across several lanes and measure vehicle speed and traffic volume. However, the main disadvantage of a non-intrusive sensor is they are highly affected by environmental conditions such as weather. Another difficulty of non-intrusive sensors such as cameras is, they are susceptible to reduced performance caused by low-light conditions. For example, detecting vehicles using cameras in night situations might be difficult compare to daylight situations. Therefore, because of the problems existing in using the camera as sensor technology in ITS, one of the aims of this project will be to improve camera utilization for object detection in night and daylight conditions.

2.1.3 Intelligent Video System

Further utilization of the camera as sensor technology in ITS is an intelligent video system (IVS). IVS is the integration of numerous computer vision and image recognition algorithms, which analyses the recognition of entity, action, and behavior. Elliot (2010) states that an intelligent video system is an automation system that utilizes the technology to perform actions based on live or stored video image images, without human intervention. IVS application on intelligent transportation systems uses cameras and sensors installed on the roads to monitor traffic conditions.

There are some publications and researches in the intelligent video system field dealing with developing a framework to provide an effective solution to interpret real-time situations and detect various events. Lee et al. (2012) proposed a framework that integrates several heterogeneous devices such as

cameras to handle different connection protocols and media encodings. Using a centralized server and public network, this framework aims to tackle one of the common problems in an intelligent video system: the lack of interoperability of network cameras to be implemented on a large scale. However, cameras that connected to public networks were vulnerable to a cyberthreat, thus applying strict access to large scale IVS is a necessity. Lee et al. (2012) also explain that the proposed framework implements the Kerberos protocol to secure captured images from threat actors' eavesdropping.

Another IVS framework proposed by Nazare Jr. and Schwartz (2016) called Smart Surveillance Framework (SSF) aims to provide scalability and flexibility of IVS. SSF provides tools that enable a user to create various projects, from small scale to large scale applications. The tools include data representation, network and communication control, parallel processing, and storage. From the real-world implementation to recognized faces of people, Nazare Jr. and Schwartz (2016) allow other research to contribute and evaluate the framework.

Furthermore, the intelligent video system must also able to identify several events at certain times. Lim et al. (2014) present iSurveillance, a smart video system framework that can obtain information in various regions-of-interest (ROI) of a video scene. iSurveillance utilizes knowledge-based architecture to increase the understanding of intricate patterns in the video scene—the presented solution counter another intelligent video system that can only perform a singular task. Also, Loredana Caruccio et al. (2019) published an IVS framework called EDCAR or Elements and Descriptors of Context and Action Representations to obtain information from video frame sequences. Similar to Lim et al. works, EDCAR use context representation and context descriptor to define the structure of actions and the order of events. Context representation is a method to represent data in machine whereas context descriptor is a method to describe feature and shape. Loredana Caruccio et al. (2019) further assert that EDCAR works by reducing the complexity of events using context descriptors, provide reasoning and information lost in the circumstances and make a conclusion based on processed data.

2.1.4 Object Detection with Deep Learning

To gain a complete understanding of traffic scenes, one should focus on classifying the different types of vehicles and trying to estimate the spatial information of an object precisely—the task to determine where objects are located in a scene or image known as object detection. As one of the fundamental computer vision problems, object detection can supply useful knowledge for the semantic interpretation of images and videos. Generally, object detection activities are divided into three stages: informative region selection, feature extraction, and classification. To obtain any information on an image, it is necessary to implement a sliding window technique to scan the whole image. One of the general approach of sliding window technique is scan the image a determined kernel with SxS dimension. However, this technique's major drawback is that it's computationally expensive since it has to produce a large number of candidate windows.

The second stage is to extract visual features, which can provide a semantic representation of an object. There are several traditional methods to perform features extractor. Lienhart and Maydt (2002) present their Haar-like features object detection model. Lowe (2004) proposed the SIFT algorithm that guarantees the extracted features have invariant features such as scaling, rotation, and translation. Although the SIFT robust to light and noise, it faces complexity problems and slow detection speed. The last stage of object detection activities is a classification in which a model should distinguish a target object from all other categories and render a target object more hierarchical, semantic, and descriptive for visual recognition. Zhang (2017) described a Support Vector Machine (SVM) used for classification and regression.

Freund and Schapire (1997), in their works, proposed AdaBoost. AdaBoost presumed that total number of samples (n) in the training set are of the same weight. For each training, the data is adjusted in the training set and increases the wrong samples' weight. The classifier will concentrate on the wrong samples. After n training rounds, n weak classifiers are combined, and corresponding weights are allocated according to each classifier's output to form a solid classifier

with high accuracy and low error rate.

Furthermore, the attempt to perform object detection using conventional methods ensures the extraction of rich and accurate features. However, the extracted features from the conventional method are typically low level and artificially selected. These methods cannot perform well on a large number of multi-class objects. Advancing deep learning on object detection can offer various degrees of detection performance enhancement and make real-time and precise object detection more workable.

Deep learning is a sub-field of machine learning that attempts to learn a hierarchy of features from input data. Deep learning algorithms can automatically learn features at several levels, allowing the algorithm to learn complex mapping functions directly from an input, without the aid of human-made features. One of the deep learning algorithms used for object detection is Convolutional Neural Network (CNN). Maried et al. (2017) emphasize that CNN is one of the most efficient deep neural networks for image processing tasks.

Similar to the classic neural network, CNN consists of layers and neurons that binding each layer together. Each layer of CNN is referred to as a feature map. The feature map's input layer is a 3D pixel intensity matrix for various color channels, e.g., RGB or HSV. The feature map of any internal layer is an induced multi-channel graphic, the 'pixel' of which can be viewed as a particular feature. Various forms of transformations can be done on feature maps, such as filtering and pooling. Filtering or convolution is an operation in which the filter matrix is based on the learned weights convolutes with the receptive neuron layer's values. The pooling process, such as max pooling, average pooling, L2-pooling, and local contrast normalization, summarises the response of the receptive field in one value to produce more robust representations of the functions.

The constructed feature hierarchy can be fine-tuned in a supervised manner by adding several Fully Connected (FC) layers to adapt to different visual tasks. Also, to get a specific conditional probability for each output neuron, the final layer with different activation functions is added. Using the mean square error or cross-entropy loss, a neural network can be optimized via the

Stochastic Gradient Descent (SGD) method. Simonyan and Zisserman (2015) present their implementation of CNN for large-scale image recognition called VGG16. VGG16 consists of thirteen convolutional layers, three fully connected layers, three max-pooling layers, and using soft-max loss as a classification layer. Convoluting 3*3 filter windows which is a dimension of filter array produce the convolutional feature maps, and the feature map resolutions are reduced with two stride max-pooling layers.

CNN is capable of extracting higher-level features and also perform sorting and classification of features in the same model. Based on these capabilities, CNN algorithms for object detection is divided into two types, classification proposal (two-stages) and regression problem algorithm (single-stage). The first one describing an object detection model to generates region proposals and classifying each proposal into different object classes. The second type focus on implement a unified object detection model to achieved final results directly. Girshick et al. (2014) present their two-stages neural network for object detection based on region proposal, called R-CNN. Instead of processing a huge number of regions, R-CNN uses selective search to obtain only 2000 candidate regions and AlexNet as a feature extractor to get the feature from the selected candidate region (Krizhevsky et al., 2012). R-CNN uses multiple SVM methods in the classification stage and fine-tunes the resulted in bounding boxes using linear regression. Despite achieved 58.5% accuracy on the VOC2007 dataset, R-CNN still performs slowly and requires 47 seconds to test each image. Thus R-CNN cannot be implemented for real-time applications.

Girshick (2015) present an improved R-CNN model to overcome the drawbacks of the previous R-CNN model, called Fast R-CNN. The aim of Fast RCNN is to achieve a faster object detection algorithm. Using a slightly different approach as its predecessor, Fast R-CNN could obtain shorter processing time since convolution operation is done only once per image. The convolutional layer is feed using an input image to produce feature maps. The feature maps are then warped into squares and using an RoI pooling layer. The proposal feature is then reshaped into a fixed size to be fed into the FC layer. Using the

SoftMax layer, the prediction for each class is produced based on the RoI feature vector. The average accuracy of Fast R-CNN is 70%, with training speed is nine times higher than R-CNN. However, Fast R-CNN is still using the Selective Search method to select candidate regions. Since it requires many calculations, when running on CPU, Fast R-CNN requires 2 seconds on average to obtain the candidate region.

To solve the slow selection of Selective Search, Ren et al. (2016) proposed Faster R-CNN. Instead of using Selective Search, Faster R-CNN embedding another neural network to predict the candidate region. Using this approach, Faster R-CNN only needs 10ms in the candidate region stage. However, despite showing good performance on detection accuracy, two stages detector is still challenging to meet real-time requirements. For example, Faster R-CNN only reaches five fps on detection speed. Another type of object detection model such as R-CNN and Faster R-CNN is proposed, which implements the regression technique to input images directly.

A two-stage detector such as R-CNN and Faster R-CNN follow the idea of implementing region proposal and classification. Hence, training two models will increase training volume and affecting the speed of training and detection. Redmon et al. (2016) proposed a You-Only-Look-Once (YOLO) algorithm to implement a single-stage detector. YOLO divided input image into S x S cells or smaller images, which each cell only responsible for detecting objects at the center of the cell. The cell is then predicting bounding boxes, confidence, and the class probability of an object. The bounding boxes having the class probability above a threshold value are selected and used to locate the image's object. YOLO could achieve 45 fps in exchange for detection accuracy that achieved 63.4% accuracy on COCO dataset compared to 73.2% of Faster R-CNN. The YOLO detector's drawback is that it struggles to detect small and unconventional objects within the image. Over the years, several improvements had been made to improve the YOLO detector, with the latest proposed by Bochkovskiy et al. (2020) called YOLOv4.

In order to combine the high detection accuracy of Faster R-CNN and fast

detection speed of YOLO, Liu et al. (2016) present their method Single Shot MultBox Detector (SSD). SSD uses high-level, and bottom-level feature maps to perform regression using multi-scale regional features since the feature maps of different layers have receptive fields of the corresponding sizes. The accuracy of SSD achieved 74.3% accuracy on COCO dataset, which is similar to Faster R-CNN and the detection speed of SSD reaches 59 FPS. Furthermore, both two-stages and single-stages could be used to perform classification and improved the application of deep learning for object detection.

2.1.5 Deep Learning with Edge Computing

Implementing computer vision on an intelligent video system to process large video sequences might introduce several challenges since standard computer vision methods utilize deep learning to process the camera's data. Deep learning is a type of machine learning that enables computers to learn from experience and understand the environment and commonly use it to solve computer vision, speech recognition, and natural language processing (Kim, 2016). One of the significant challenges is to satisfy the high computing requirements of deep learning with an intelligent video system's resources. The installation of a high computation device on the roads will be inefficient since it will requires similar infrastructure to support the device. However, moving the processing system to a centralized location might also ineffective since the data need to be processed in real-time.

Chen and Ran (2019) explain that three things should be considered in implementing deep learning in a centralized location. First, latency or time interval of communications. Deep learning applications on computer vision required real-time inference. Thus sending to a centralized location for processing is inefficient since it may incur propagation delay and did not meet the low-latency requirements of deep learning. Second, scalability or the ability of the system to adapt to the increasing of connected devices. The large numbers of connected devices will also increase the data that needs to be uploaded. Hence,

moving the processing system to a centralized location will raise a concern. Furthermore, sending data to the centralized location that located far from data sources will increase privacy concern. The data that needs to be uploaded might contain sensitive information and will attract threat actors. Chen and Ran (2019) further suggest that edge computing is a feasible option for solving latency, scalability, and privacy challenges during the implementation of an intelligent video system. In edge computing, integrated computation resources provide data processing abilities close to the end service (Satyanarayanan et al., 2009). Many research efforts have been conducted to address latency challenges and meet the computational requirements of deep learning. Taylor et al. (2018) present their results to achieve the desired accuracy and inference time of the Deep Neural Network (DNN) implementation, focusing on the image classification task. In their research, Taylor et al. use Jetson TX2, a GPU-based embedded system, to speed up deep learning inference. Based on the evaluation metrics, Taylor et al.'s research achieved a 7.52% improvement in accuracy metric and 1.8x reduction in inference time compared to the other deep learning model. Lai and Suda (2018) investigate deep learning implementation on resource-limited hardware. In their research, Lai and Suda introduce CMSIS – NN, a software library to enable deep learning on microcontrollers with limited to compute resources, memory, and power. Nikouei et al.'s (2018) research aim to reduce the time delay of video processing application. Thus they proposed the implementation of a deep neural network to detect a pedestrian. Nikouei et al. evaluated their proposed deep neural network on Raspberry Pi 3 and achieved an affordable computation workload. Furthermore, several works have been described to accelerate the implementation of deep learning across end devices.

2.2 Related Works

This section covers some of the implementation works related to the proposed intelligent video system.

2.2.1 Streaming Pipeline

Implementing deep learning on edge computing become more complex since the neural network may process multiple input streams simultaneously. In this project, the proposed stream processing paradigm is used to address the complexity of the streaming pipeline. Stephens (1997) explains that stream processing is a computer science term to describe a number of disparate systems such as the dataflow system, where the output of one module inside the streaming pipeline is provided as the input to another module. There are several studies related to the development of stream processing. Lin and Tang (2011) proposed their stream processing to perform moving object detection. The proposed framework use filter modules to define the process of their stream processing. The source filter has a function to handle data obtainment. The transform filter is responsible for data processing and conversion. The rendering filter responsible for the ultimate flow of the data. Zhou et al. (2015) present their stream processing architecture to detect traffic signs. The proposed architecture is based on a field-programmable gate array (FPGA) and consists of 5 modules with the aim to perform classification based on an input image. Ren et al. (2018) present Traffic Camera Pipeline (TCP), a stream processing framework that performs object detection from the video stream. TCP performs five steps to obtain trajectories from detected objects, which are including, traffic video collection, perform object detection using deep learning, object labeling, image transformation, and vehicle trajectories extraction. Another study, presented by Dong et al. (2013), studied how Dynamic Parallelism using GPU can accelerate the performance of stream processing. Dong et al. show that their approach achieved 154 times faster than the implementation of similar stream processing using CPU. Different from existing solutions and studies, the proposed intelligent video system leverages stream processing implementation using six filter modules on a GPU-based embedded device.

2.2.2 Object Detection in Traffic Scenes using Deep Learning

The recent development of object detection using deep learning to extract traffic information attract many types of research and studies. Peng et al. (2020) present their experiment using combined object detection and semantic segmentation to obtain information from traffic scenes. Fang et al. (2016) present a novel multiple channel feature called Deep Compact Channel Feature (DCCF) to generates discriminative feature representation and use it to perform traffic signs detection. Another study, presented by Jeong et al. (2019), studied the combination of real-time object detection and situation recognition. They proposed YOLO as a detector to track objects and Long Short-term memory to extract object information. Using this approach, object behavior such as when a pedestrian walks or stood on the sidewalk could be obtained. In addition, Ye et al. (2020) proposed a novel SSD framework called feature-enhanced SSD (FE-SSD) to address challenges in railway object detection, which has relatively small accuracy for detecting small objects. This project aims to compare existing successful deep learning algorithms and make appropriate improvements, including developing integration method of the deep learning to the end-to-end stream processing.

2.2.3 Object Re-Identification using Deep Learning

Object re-identification is one of the challenging issues in an intelligent video system. Object re-identification is the ability of a system to detect the object in different imaging conditions such as lighting conditions and object pose. Geng et al. (2016) proposed a pedestrian re-identification algorithm that used several loss functions to train the network, including classification and verification loss. In their approach, Geng et al. present the method to judge whether two images belong to the same pedestrian. Lin et al. (2019) proposed pedestrian re-identification based on multiple attributes. The proposed model predicts the identification of information and attributes of each pedestrian. Another study, presented by Li et al. (2017), studied how a joint deep learning model could effectively extract discriminative representations for vehicle images. Using their

novel approach, Li et al. demonstrated the ability of vehicle re-identification and image retrieval. In addition, Zhang et al. (2017) present the novel approach for vehicle re-identification using a guided Triplet network to improve re-identification efficiency. This project leverages existing vehicle re-identification methods and proposed improved loss functions using joint triplet-loss and cross-entropy.

Chapter 3

System Design

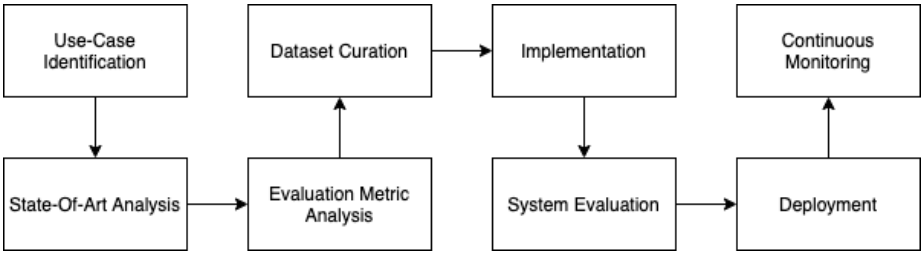

Figure 3.1: AddInsight-Vision Development Framework.

In the recent development of intelligent video systems using computer vision and deep learning, researchers should still address open issues that should be addressed by researchers, including biased output, dataset creation, and the effectiveness of the system. As shown in Figure 3.1 , this project will present a development technique to address the challenge of building an intelligent video system. The use-case of AddInsight-Vision is to obtain traffic information using computer vision and deep learning to perform object detection in traffic scenes. To achieve the aim of AddInsight Vision, the state-of-art analysis is performed to evaluate the best-suited technology and framework. The following are state-of-art methods used in this project: stream processing paradigm to create an effective and efficient intelligent video system based on modular filters, implement fast and accurate object detection, develop vehicle re-identification model using improved loss function, and implementation of deep learning on edge computing using GPU-based embedded device.

Inherently, the deep learning model used for object detection depends on the evaluation metric to measure the detector's effectiveness. In this project, two different methods are used to quantify the system's performance, especially

for object detection model and vehicle re-identification. The evaluation metrics are Mean Average Precision (MAP) and Cumulative Matching Characteristic (CMC). In addition, the performance of the system also depended on the dataset used for training and evaluation. Choosing the right dataset for the right purpose will guarantee the system's quality and avoid the bias of the outcomes. In this project, two datasets will be used to evaluate the system. First, the dataset from the South Australian Department of Planning, Transport, and Infrastructure (DPTI) consists of 6 hours of long traffic scenes, and second, the Ve-Ri dataset to evaluate the performance of the vehicle re-identification model.

The next process of developing AddInsight Vision is to conduct implementation and evaluation, including developing stream processing, training and evaluating the models based on the metrics and testing to measure the qualitative performance. The last process of AddInsight-Vision development is the deployment and continuous monitoring of the system. AddInsight-Vision is planned to be deployed to the City of Launceston in Tasmania to measure its performance with real-time conditions.

Furthermore, in this chapter, section 3.1 will discuss the strategies to create an effective intelligent video system, including the methodology of the project development and how it will be integrated with the current AddInsight System. Section 3.3 will detail the approach to increase the performance of the IVS using the Transfer Learning technique. Also, section 3.4 will present the details of improved loss function for vehicle re-identification.

3.1 Stream Processing Architecture of AddInsight-Vision

Figure 3.2 shows an overview of the proposed framework of AddInsight-Vision using the stream processing paradigm. This project uses Deepstream as a framework for the streaming pipeline. Deepstream is an accelerated deep learning framework to build an intelligent video system (NVIDIA, 2020). The reason for using Deepstream in this project is that it provides neural network filters and supports a conventional video stream. In addition, Deepstream allows the

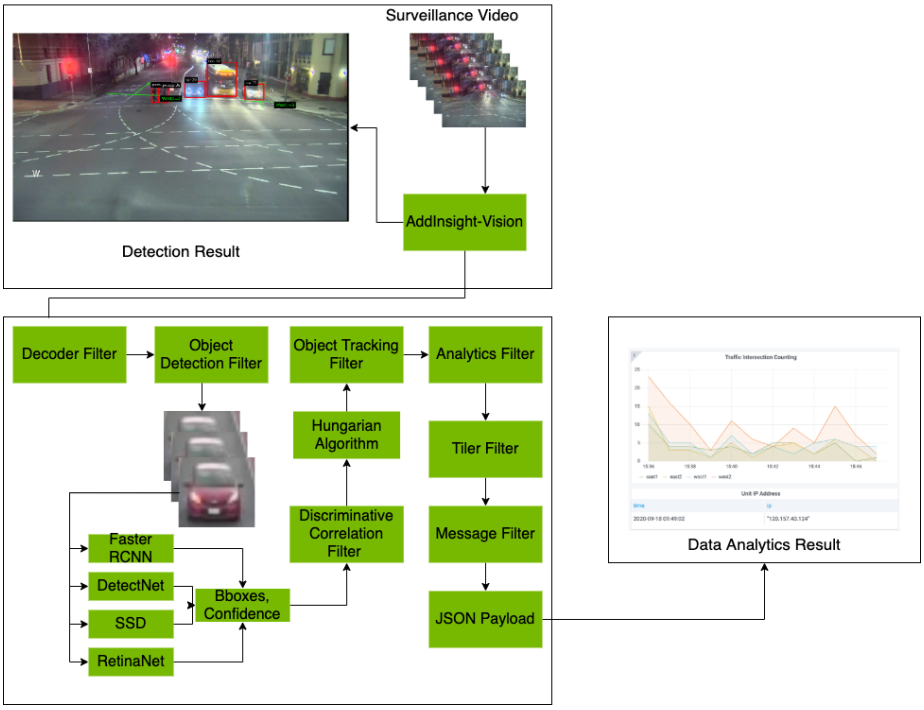

Figure 3.2: AddInsight-Vision Pipeline.

pipeline to be implemented on an edge computing device, which is one of the aims of this project. The AddInsight-Vision architecture consists of six stream filters. The first filter is the decoder filter, which is the filter used to convert video sequences into a stream data format. Deepstream uses a standard, extensible structure of metadata called NvDsBatchMeta to allow data flow in the pipeline. The second filter is an object detection filter used to perform inference from the input sequence. In this project, four different object detection algorithms are used, including RetinaNet, Detectnet, SSD, and Faster R-CNN. Details of each algorithm will be described in section 3.2. The third filter is an object tracking filter that tracks the detected object and gives each new object a unique ID. This project utilized a discriminative correlation filter (DCF) based approach for visual object tracking and a Hungarian algorithm for data association.

Several analytics functions are applied in the analytics filter, including the

Region of Interest (ROI) filtering, Direction Detection, and Line Crossing. The metadata from the analytics filter is then processed using a Tiler filter is the fifth to combine several streams input and encode stream data into a video format for on-screen display purposes. The six filter of the AddInsight-Vision pipeline is a message filter that contains a message broker and a message converter. A message broker is used to communicate with the message stream platform, whereas the message converter converts the stream data into JSON payload. The message filter's function is to perform data exchange with the AddInsight dashboard where the data, such as total vehicles detected, from the AddInsight, is transferred to the database for further analytics. Besides using the six-filter described in this section, the stream processing framework of AddInsight-Vision also using another filter such as a multiplexer filter to combine several metadata and on-screen display filter. Details of AddInsight stream-processing will be discussed in chapter 4.

3.2 AddInsight-Vision Edge Computing Infrastructure

In order to implement AddInsight-Vision on an edge device, a major challenge is to choose the right device that meets hardware requirements to run deep learning models. In this project, a Graphics Processing Unit (GPU) based embedded device called Jetson AGX Xavier (Figure 3.3) is proposed to run the AddInsight-Vision framework ("Deploy AI-Powered Autonomous Machines at Scale," 2020). Jetson AGX spesification meet the requirement of implementing deep learning on Edge. Jetson AGX Xavier has 64 Tensor Cores GPU and support of Deep Learning Accelerator that able to run a deep learning model until 32 TOPS (Tera Operation per Second). Another feature of Jetson AGX is Vision Accelerator, which is a feature that could use to optimize multi-stream processing. Table 3.1 describes the details of Jetson AGX Xavier's specifications. Furthermore, several peripherals (Table 3.2), including power supply, IP camera, and 4G modem, are installed together to support the AddInsight-Vision edge computing infrastructure.

Image removed due to copyright restriction.

Figure 3.3: Jetson AGX Xavier (NVIDIA 2020).

Item	**Description**
CPU	8 Core Carmel CPU @ 2133 MHz
GPU	512 Core Volta @ 1.37 GHz
DL Accelerator	2x NVDLA
Vision Accelerator	7-Way VLIW Processor
Memory	16GB LPDDR4X
Storage	32 GB
Power	10 W / 15 W / 30 W
Video Encode	4kp60, 4Kp30
Video Decode	8Kp30, 4Kp30

Table 3.1: Hardware Specification of Jetson AGX Xavier.

Item	**Model**	**Power Consumption**
Processing Unit	Jetson AGX Xavier	Max 30 W
IP Camera	Axis P1357-E	40 W
4G Modem	Classified	Classified
Switch	Classified	Classified
Power Supply	Classified	Classified

Table 3.2: Jetson AGX Xavier Edge Infrastructure.

3.3 AddInsight-Vision Object Detection Model

3.3.1 Vision Information Extraction

In this project, four separate object detection algorithms are tested to measure their output by collecting traffic information. The object detection algorithm is based on the Convolution Neural Network (CNN). CNN is a deep learning algorithm that can capture the input image, attach value to different objects in the image, and be able to identify each object in a class or category. The architecture of CNN is similar to the pattern of connectivity of neurons in the human brain and has been influenced by the organization of the visual cortex. Individual neurons respond to a stimulus only in a limited area of the visual field known as the Receptive Field. The array of these fields overlaps to fill the entire visual region. Figure 3.4 described the architecture of a CNN.

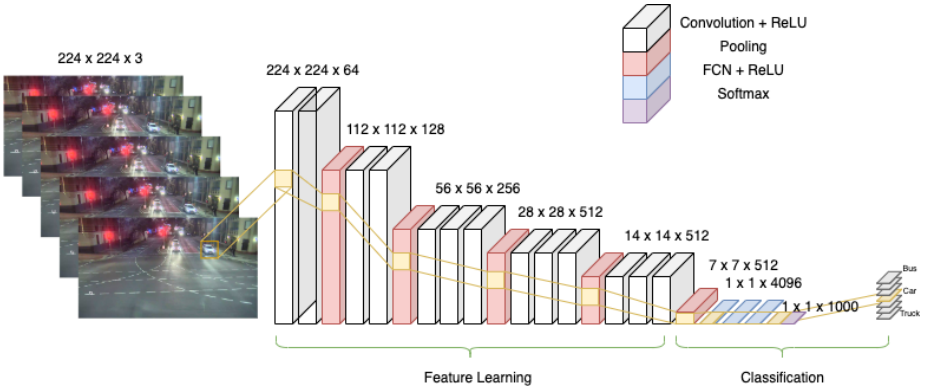

Figure 3.4: Architecture of Convolutional Neural Network.

In contrast to the traditional neural network, CNN is constructed with several convolution hidden layers. The function of these hidden layers is to capture complex spatial and temporal dependencies of an input, which cannot be done using a traditional neural network. CNN consists of two stacks of layers to produce an object's prediction based on the input image. The first group of layers is a feature learning group that functions to extract features from the input image. In this first group, it consists of a convolution layer and a pooling

layer. The purpose of the convolution layer is to extract high-level features from the input image, which is, in this case, using convolution. Based on Figure 3.4, which shows the architecture of VGG16, there is a total of 13 convolutional layers in the VGG16 architecture. The rectified Linear Unit (ReLU) activation function is added to the convolutional layer. The activation function is used to determine the output of the layer, whether it passed the input to the next layer or not.

The first two layers of VGG16 use 64 filters. The input image that passed through these layers is multiplied with 3×3 kernels with a stride of 1, which means the kernel is shifting nine times every time the multiplication operation between the kernel and the portion of the input image. The result from the first two layers is the feature map matrix with $224 \times 224 \times 64$ dimensions. In the VGG architecture, the Pooling layer is added between the convolutional layer used to segregate the features map's dominant features and reduce the computational power required to process the data. VGG16 utilized max-pooling, which reduces the dimension by only returns the maximum value of feature maps. The next group of the CNN architecture is classification layers. This group consists of fully connected layers and loss function to take the output from the feature learning layer and use it to classify the image into a label. In the VGG16, the output from the feature learning group is then flattened into a column with a $1 \times 1 \times 4096$ dimension. The flattened vector is fed into a full backpropagation network to determine the most accurate weights and uses a SoftMax classifier function to determine the input class based on its weight values. In particular, the feature extractor used in this project is based on ResNet-18. ResNet-18 represents a good trade-off between computational time and performance (He et al., 2015). Table 3.3 describe the Resnet-18 architecture.

ResNet-18 is known for solving the vanishing gradient problem on a neural network. Vanishing gradient problem occurred when the network is too deep, and the gradients from where the loss function is calculated easily shrink to zero after multiple training. This caused the training weight to never updating its value. Thus no learning is being performed. By looking at the output of

Layer	**Output Size**	**ResNet-18**
conv1	$112 \times 512 \times 64$	7×7,64,stride 2
conv2_x	$56 \times 56 \times 64$	$3 \times 3, stride2$
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
con5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7average pool
fully connected	1000	512×1000fully connections
softmax	1000	

Table 3.3: ResNet-18 Architecture.

ResNet-18, the network produced 512 dimensions fully connected layer vector, which contains a lot of information about the image and will be used for object detection algorithm.

The first detector to be tested in this project is DetectNet, a single-stage detector. NVIDIA (2016) suggested that DetectNet overcome one of the training processes relevant to the marking of training results. Training data samples are larger images containing a variety of objects. For each object in the image, the training mark must capture the class of the object and the coordinates of the corners of the bounding box. While the number of objects will vary between training images, a simple choice of mark formats of various lengths and dimensions would make it difficult to define the loss function. DetectNet solves this main issue by introducing a set 3-dimensional mark format that allows DetectNet to absorb any size images with a variable number of objects present. Figure 3.5 shows the architecture of DetectNet.

Three important processes of DetectNet are described in Figure 3.5. First, the first two layers of DetectNet perform data ingestion and data augmentation. Data ingestion is the process to feed the data to the network and data augmentitation is the process of modify the data in order to meet with network specification.

Figure 3.5: Architecture of DetectNet V2 (NVIDIA 2016).

Second, a fully-convolutional network (FCN) performs feature extraction, predicts object classes, and produces bounding boxes. In this project, ResNet-18 is used as the FCN of DetectNet. The third process is the loss function, which is a function that simultaneously measures the error in the two tasks of predicting the object class and bounding boxes. DetectNet uses a linear combination of two separates loss, namely, coverage_loss and bbox_loss functions, to produce its final loss function that will be used for network optimization. Coverage_loss as defined in equation 3.1 is the sum squares of differences between the true and predicted object coverage across all grid squares in a training data sample.

$$\frac{1}{2N}\sum_{i=1}^{N}|coverage_i^t - coverage_i^p|^2 \tag{3.1}$$

Whereas, bbox_loss (equation 3.2) is the mean absolute loss to measure the bounding boxes' true and predicted corners.

$$\frac{1}{2N}\sum_{i=1}^{N}[|x_1^t - x_1^p| + |y_1^t - y_1^p| + |x_2^t - x_2^p| + |y_2^t - y_2^p|] \tag{3.2}$$

The advantage of utilizing DetectNet for object detection is the efficiency with which all objects inside a large image can be detected and produce accurate bounding boxes prediction.

Another detector evaluated in this project is RetinaNet. Similar to DetectNet, RetinaNet is a single-stage object detection algorithm. Lin et al. (2018) proposed RetinaNet to address the accuracy of a single-stage detector due to foreground-background class imbalance. As described in Figure 3.6, RetinaNet consists of a

feature extractor using ResNet and Feature Pyramid Network (FPN) and two task-specific subnetworks for classification and bounding box regression.

Image removed due to copyright restriction.

Figure 3.6: Architecture of RetinaNet (Lin et al. 2018).

Feature Pyramid Network in RetinaNet is a tool for generating feature maps using a featured image pyramid technique. A featured image pyramid is a method for sub-sample images captured in lower resolution and smaller images, thereby executing a stack of images. Using this method, the production size of the feature maps can be minimized. In addition, RetinaNet uses focal loss to calculate the model error and is implemented to address the class imbalance issue with single-stage object detection models. As there can be thousands of anchor boxes in RetinaNet, only a few will be allocated to the ground-truth object. Although generating minor losses, high-probability anchor boxes could collectively overpower the network. The losses from unassigned anchor boxes would be minimized by focal loss. Figure 3.7 shows the comparison of RetinaNet and another object detector such YOLO and SSD in terms of inference time and accuracy. RetinaNet could achieved fast inference speed while maintaining the detection accuracy.

Another two detectors that will be evaluated in this project are the Single Shot Detector (SSD) and two-stage detector Faster R-CNN. However, unlike the previous explanation, in this project, the feature extraction backbone for SSD and Faster R-CNN will be based on ResNet-18. ResNet-18 is used since it provides a light training process and does not require a high computation device. Furthermore, each detector with ResNet-18 as a feature extractor will be trained to using the Transfer Learning technique to increase the accuracy and inference

Image removed due to copyright restriction.

Figure 3.7: Baseline of RetinaNet (Lin et al. 2018).

speed.

3.3.2 Dataset Curation

Data curation is the process of choosing the most relevant data to meet the system needs on a specific function. Data curation is an important step in this project since raw data such as traffic scenes video is unusable to train the deep learning models. To clarify, data curation in this project is the process of transforming raw data into a specific data type that could be used to train the object detection and vehicle re-identification model to increase the accuracy of the trained deep learning models. The process of data curation is described in figure below. Based on Figure 3.8, the curation process's high-level steps are

Figure 3.8: Dataset Curation Process.

first to transform the data type to a specific data format required by the model. In this project, the raw data comes from the traffic scenes video provided by the South Australian Department of Planning, Transport, and Infrastructure (DPTI). The DPTI videos are recorded surveillance videos from two East Terrace and Grenfell Street intersection cameras at a specific time, including 08.00 AM, 12.00 PM, and 07.00 PM. The resolution of the video is 1920×1080 pixels, with the duration of each video is one hour. To create a dataset that meets

the training and evaluation requirements, the video is transformed into a JPEG image sequence with dimension 480×288 pixels. The dimension is chosen based on the capability of device in transfer learning process. Larger dimension will require high computational process. The next step of data curation is performing data labeling on the transformed data. Data labeling is the process of determining the object of interest in an image. For object detection, the data labeling process is the process to determine the ground truth of each object by drawing a rectangle on the area of the object and saving the pixel coordinates of the rectangle, and tag it to the image. In this project, data labeling tools called CVAT (Figure 3.9) is used to perform manual labeling of the transformed data.

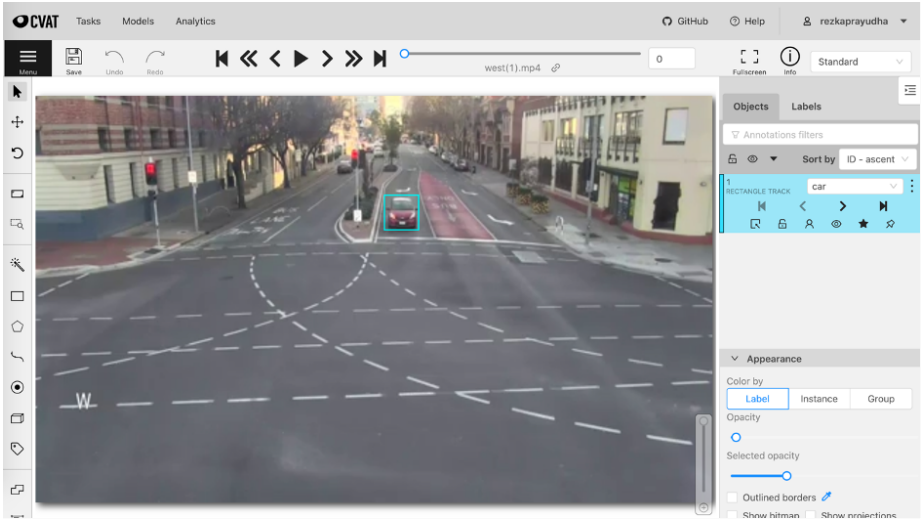

Figure 3.9: CVAT Annotation Tool

The next step of data curation is constructing a dataset. The dataset format used in this project is the KITTI format (Geiger et al., 2013). The directory structure of the KITTI format is described in Table 3.4. Each image in the dataset is pair with one label that contains information of objects in the image, such as type of objects and coordinates of bounding boxes. The structure of the label (frame00001.txt) described as: Frame Name | Track ID | Type | Truncated | Occluded | Alpha | BBox 3D | Dimension | Location | Rotation.

Image Folder	Label Folder
frame00001.jpeg	frame00001.txt

Table 3.4: KITTI Dataset Directory Structure.

3.3.3 Transfer Learning Technique

To increase the object detection models' accuracy for a specific purpose, the models should be trained using a specific dataset. For example, to detect buses, the object detection models should be trained using a dataset that contains the bus. However, training an object detection model requires a lot of resources and time. (You et al., 2018) reveal that training the ResNet-50 neural network based on the ImageNet dataset using Nvidia M40 GPU required 14 days. Moreover, the object detection process continuously shifts as new types of objects are found. In this project, AddInsight vision tries to detect a specific bus that is not supported by pre-trained object detection models. Furthermore, using transfer learning, one could increase object detection models' performance and overcome the extensive time of the training process.

Transfer learning is a method to create new specific training and speed up the process using previous training information. Tan et al. (2018) explain transfer learning as the process to transfer knowledge from the source domain to the target domain. In Source domain is represented as $D = x, P(X)$ with x as feature space and $P(X)$ as probability distribution where $x = x1, x2, ..xn \in x$. The domain task is represented as $T = y, f(x)$, with y as target label, and $f(x)$ as target predicted function. Using this scenario, Tan et al. described that, in the condition of T_T based on D_T, D_S could improve the learning task of T_S. Transfer learning aims to improve predictive function $f_{T(.)}$ for learning task T_T using knowledge from D_S and T_S. In this case, $Ds \neq Dt$ and/or $T_S \neq T_T$ and the size of $D_S >> D_T$.

Tan et al. (2018) also explain four approach categories of transfer learning, including instance-based, mapping-based, network-based, and adversarial-based (Figure 3.10). In this project, the network-based approach will be used to train

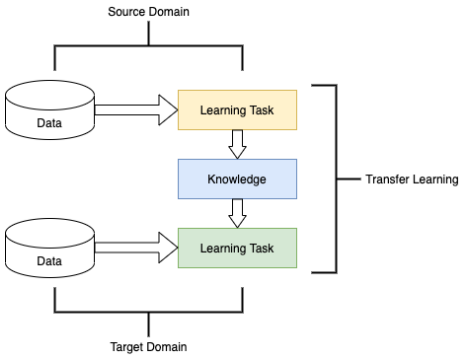

Figure 3.10: Transfer Learning Process (adapted from Tan et al. 2018).

the object detection model. Network-based transfer learning is the form of transfer learning which using the pre-trained network in the source domain. In the neural network field, a pre-trained model means a model that has been trained on a specific dataset with a different number of classes. Network-based transfer learning refers to using the neural network architecture, re-purpose the output for specific needs. Since the neural network architecture consists of feature learning part and classification part, there are three strategies (Figure 3.11) to do the network-based transfer learning.

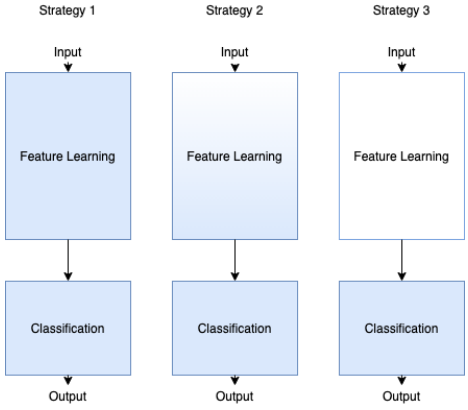

Figure 3.11: Transfer Learning Strategy.

In the first strategy, the entire network is used to train the dataset. The

major advantage of using this strategy is, it requires a large dataset and high computational resources. The second strategy is partially training the network architecture. In this strategy, one should specifically choose which part of the network should be trained and left frozen. If the task required lots of parameters to be trained based on a small dataset, the number of frozen layers should be larger than train layers to avoid overfitting. The last strategy is only the classification layers trained or fine-tune and use the feature learning layers to only feed the classification layers. This approach is useful if the dataset is small, and the pre-trained network has a similar task with the target domain task. In this project, a four-quadrant model is used to determine the best strategy to do network-based transfer learning (Figure 3.12). In this project, the dataset

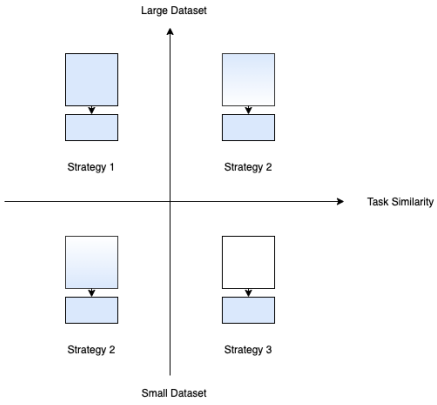

Figure 3.12: Four-quadrant Model of Transfer Learning Strategy.

contains 7180 images and 7180 labels, which is relatively small to use strategy 1. Moreover, based on the small dataset condition and this project's task, which is to perform object detection, the best strategy for transfer learning is strategy 3. In strategy 3, the aim is to modify the classification layers, including determining the class of objects and the training parameter. Four pre-trained models are used for transfer learning, namely, DetectNet, SSD, RetinaNet, and Faster R-CNN. The pre-trained models used in this project have been trained on the Open Image dataset, consisting of almost 9 million images (Kuznetsova et al., 2020).

3.4 AddInsight-Vision Vehicle Re-Identification Model

This project also extends to add a vehicle re-identification model to the Add-Insight framework. Vehicle re-identification (Re-ID) aims to identify and track the same vehicle from multiple cameras. However, one of the obstacles using a camera as a data source to re-identify the camera is difficult due to the relatively same feature of the vehicle. Watcharapinchai and Rujikietgumjorn (2017) proposed a method of vehicle re-identification using license plate matching. Nevertheless, in most video surveillance, the license plate is hardly recognized. Furthermore, using the Convolutional Neural Network (CNN) to learn the vehicle's feature, an improved vehicle re-identification method is proposed.

CNN is used due to the ability to extract deep features of an image. In addition, CNN also able to perform the classification of an object based on its class. In this project, a proposed vehicle Re-ID method is based on feature extraction and classification based on image similarity. This method will focus on extracting spatiotemporal information from the vehicle images. Spatiotemporal information is a type of information related to a certain location and time. For example, vehicle A may appear in camera X at 08.00 AM and appear in camera Y at 09.00 AM. The spatiotemporal information of vehicle A is:

$$VehicleA|camerax08.00am|cameray09.00am$$

Liu et al. (2016) present a Ve-RI dataset, which is a large vehicle re-identification dataset that contains spatiotemporal information. VeRi dataset consists of 40.395 images that come from 20 cameras, recording an area of 1.0km2. The significant part of the VeRi dataset is the cross-camera vehicle correlation that provides spatiotemporal information, including camera ID and timestamp. In this project, the Ve-RI dataset is divided into training and testing datasets, with total images for training is 37,778 and 11,579 for testing. Furthermore, the proposed vehicle Re-ID method will be evaluated using the VeRI dataset.

The proposed method for vehicle Re-ID is carried out in the framework below (Figure 3.14, including data preparation, feature learning model, and evaluation

Image removed due to copyright restriction.

Figure 3.13: Ve-Ri Dataset (Liu et al. 2016).

using the re-rank method.

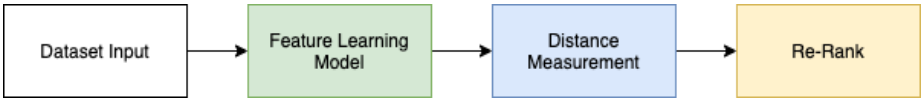

Figure 3.14: Vehicle Re-Identification (Re-ID).

The first step of vehicle re-ID is preparing the input data, including dataset augmentation, including image transformation, in order to increase the diversity of the data. The diversity of the data is important in the training process to avoid overfitting and biased of the training outcomes. There are three dataset directories of the Ve-RI dataset that will be used in the process, which are the training dataset, testing dataset, and query dataset. The training dataset is used to train the vehicle Re-ID model. The training dataset consists of a positive and negative pair of images that will be used for triplet sample construction of the vehicle-Re-ID model. Positive image is the image that has similar feature with the image inside testing folder, whereas the negative image have distinct feature with the testing image. The image inside query dataset will be used to measure the accuracy of each image with the images inside the testing dataset. The augmented dataset is then parsing to the feature learning model based on ResNet-50 architecture. The difference between Resnet50 and ResNet-18 is the size and the total number of layer used in the network. The architecture of ResNet-50 is described in the table below.

Layer	**Output Size**	**ResNet-50**
conv1	$112 \times 512 \times 64$	7×7,64,stride 2
conv2_x	$56 \times 56 \times 64$	$3 \times 3, stride2$
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
con5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
average pool	$1 \times 1 \times 512$	7×7average pool
fully connected	1000	512×1000fully connections
softmax	1000	

Table 3.5: ResNet-50 Architecture.

In this stage, a strategy two transfer learning is used to train the Ve-RI dataset in order to increase the effectiveness and reduce training time compared to retraining the entire network. Strategy 2 is picked since the Ve-RI dataset is relatively large (40000 images) and the ResNet-50. The pre-trained network has a similar task with vehicle re-identification, which is feature extraction. Using strategy 2, only the classification group of the ResNet-50 will be modified to perform vehicle re-identification, which is the loss function.

The loss function or cost function is the part of classification layers to determine the CNN model's correctness. Generally, the SoftMax classifier is used to determine the target label's correctness in the image classification problem. However, the SoftMax classifier alone could not be used for re-identification purposes since the data input, which is the vehicle images, has different features,

and is relatively difficult to distinguish only by label correction. In addition, perform a re-identification problem using only ground truth labels (0 and 1) may increase the loss if the target label is incorrect.

$$L = -(ylog(p) + (1 - y)log(1 - p)) \tag{3.3}$$

Equation 3.3 calculate the SoftMax loss of CNN. With p is the probability of feature map and y is the ground truth label 0 and 1. When the probability is high, i.e., 0.99, the loss will approximately close to zero. However, if the target label is incorrect, the loss value will be high, creating a problem in the training process. Therefore, this project proposed an improved loss function, triplet-sampling loss, and cross-entropy loss via label smoothing to regularize the loss in the classification layers.

Triplet-sampling loss is used to solve the problem in vehicle re-identification. The target domain category is flexible, such as the same vehicle images captured from a different camera. Triplet-sampling loss is the method to find the closest embedding between pairs, assuming that images with the same label must have similar embedding in the embedding space. In contrast, images that have a different label, their embedding in embedding space must be distant. The advantage of triplet-sampling loss is to distinguish the features in domain space and label space. Figure 3.15 describes the proposed triplet-sampling loss.

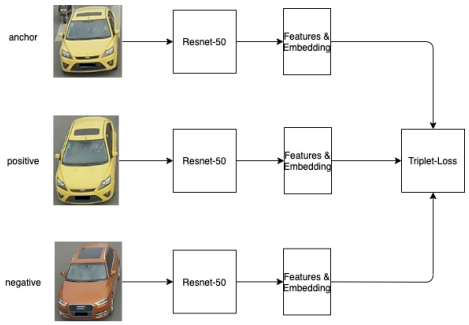

Figure 3.15: Triple-Loss Process.

The below equation is used to determine the distance between the three pairs

of images and obtain the loss function.

$$L_{htri} = max(D(A, P) - D(A, N) + margin, 0) \tag{3.4}$$

With A is an anchor or the target label, and P is the positive label, which is the label that has the same class as the anchor. $N$is the negative label of a different class. To minimize the L or Loss, using Euclidian distance function D, the $D(A, P)$ is pushed to be zero, and $D(A, N)$ should be greater than $D(A, P)$ + margin. The margin value is used to ensure that if d(a,n) is distant, the function will diminish the effort to pair the triplets, thus decrease the process of triplet-sampling. The triplet-loss function is extremely useful in finding similar images. However, using a large dataset, it is relatively difficult to produce valid triplets. Therefore, Hermans et al. (2017) suggest batch hard triplet mining as the best strategy to determine a valid triple for the triplet-loss function. A batch of embeddings B from a batch of B inputs, for a B embedding, consists of batch features of images and target label $B = PK$. For each target label or anchor, compute the biggest distance $D(A, P)$ and biggest distance $D(A, N)$. Hence, it produces a valid triplet for the triplet-sampling loss function.

The second loss function applied in the vehicle re-identification model is a cross-entropy loss.

$$L_{xent} = -\sum_{i=1}^{p} log(p(i)q(i)) \tag{3.5}$$

Given p, the probe image's probability belongs to vehicle i, and q is the ground truth label. The cross-entropy loss tries to calculate categorical cross-entropy of p and q. However, to minimize the effect of label dropout, the q distribution is replacing with uniform distribution (Szegedy et al., 2015). Hence, the q distribution function is

$$q'(k|x) = (1 - \epsilon)\delta_{k,y} + \epsilon u(k) \tag{3.6}$$

Furthermore, using triplet-sampling loss and cross-entropy loss, this project proposed an improved loss function called joint loss function that will be used for vehicle re-identification. The final joint loss function can be formulated as:

$$L_{joint} = L_{xent} + L_{htri} \tag{3.7}$$

The last step of the vehicle re-identification process is re-rank to determine the evaluate the performance of the re-identification model. In this step, using the VeRI dataset, there are 1678 images designed as a query dataset. For each image in the query dataset, the model tries to find similar images in the testing dataset. Thus each of the query images is re-rank to determine the performance to re-identifying a similar image (Figure 3.16).

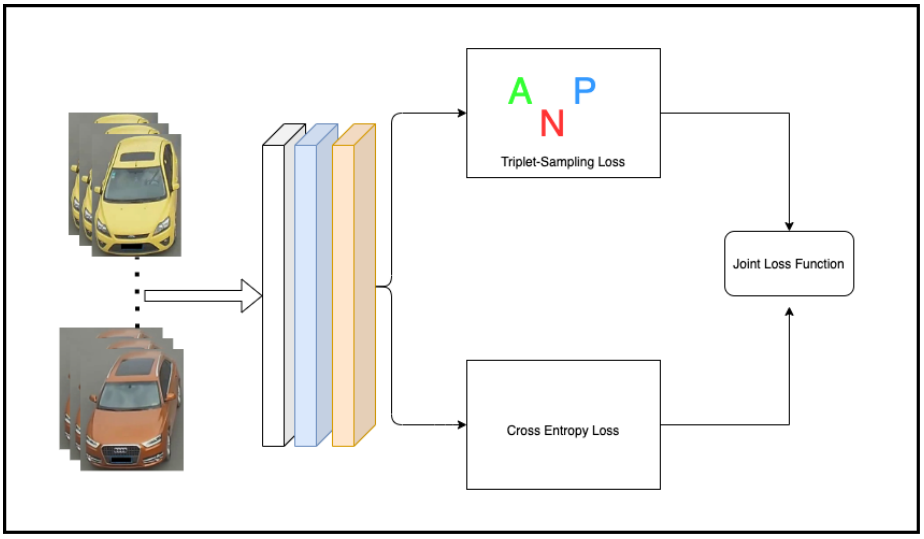

Figure 3.16: Joint Loss Function of Vehicle Re-ID.

Chapter 4

System Evaluation

This chapter will go through the experiment and evaluation of AddInsight-Vision, including the system setup and infrastructure development, followed by qualitative and quantitative analysis of the results.

4.1 AddInsight-Vision System Integration

In this project, an experiment with AddInsight-Vision is conducted using Jetson AGX Xavier. Following the structure of Figure 2, this experiment aims to perform object detection and send the data to the AddInsight Dashboard. The initial setup of the AddInsight-Vision experiment is described in the following list.

1. Jetson AGX Setup

   The Jetson AGX is installed with Jetpack 4.4 Operating System and Deepstream 5.0 Software Development Kit (SDK). To maximize the performance of AddInsight-Vision, the power mode of Jetson AGX is set to MAX (30 Watt). The Deepstream SDK used in this experiment is based on Python 3.7.

2. Input Data

   The data used for the experiment is based on surveillance videos from the South Australian Department of Planning, Transport, and Infrastructure. The videos included intersection scenes between 08.00 AM to 09.00 PM, with total of twelve videos.

3. Object Detection Model

   In the initial experiment, the object detection model is DetectNet V2 from

NVIDIA. The model has not been used for transfer learning.

4. Message Broker Server

   A message broker server called Kafka is used to accommodate information exchange between AddInsight-Vision and AddInsight-Dashboard.

The below command is used to run the AddInsight-Vision streaming pipeline.

```
$python3 deepstream-pipeline file:///[videofile_path]
```

This command will run the AddInsight-Vision and read the video input based on the described file location. The AddInsight is also able to read the input from the Real Time Streaming Protocol (RTSP) stream.

```
$python3 deepstream-pipeline rtsp://[rtsp_url]
```

AddInsight-Vision program structure consists of several elements, which are source element or the input data, manipulation elements or the filters, and destination element or sink. In order to communicate between elements, the program uses several semantic layers, namely, bin, pad, and bus. Figure 4.1 described the structure of the AddInsight-Vision program structure.

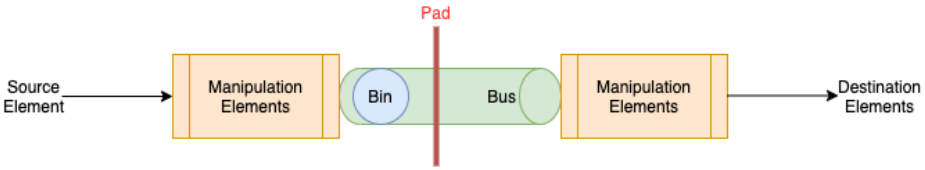

Figure 4.1: Deepstream Pipeline Structure.

In the first stage, the program reads input data from the video source and convert it into Deepstream metadata, and stores it in the bin. Below is the partial source code that describes input data reading and conversion of the input data to metadata.

```
for i in range(number_sources):
        print("Creating source_bin ",i," \n ")
        saved_count["stream_"+str(i)]=0
        uri_name=args[i+1]
```

```
        if uri_name.find("rtsp://") == 0 :
            is_live = True
        source_bin=create_source_bin(i, uri_name)
        if not source_bin:
            sys.stderr.write("Unable to create source bin \n")
        pipeline.add(source_bin)
        padname="sink_%u" %i
        sinkpad= streammux.get_request_pad(padname)
        if not sinkpad:
            sys.stderr.write("Unable to create sink pad bin \n")
        srcpad=source_bin.get_static_pad("src")
        if not srcpad:
            sys.stderr.write("Unable to create src pad bin \n")
        srcpad.link(sinkpad)
```

Based on the description, each of the elements in the pipeline is linked with a pad. Pad is like a "gate" that determine which data type could pass through the element. The metadata inside the bin is then pass-through manipulation elements or the filters. The filters are linked together inside the pipeline to process the metadata.

```
print("Adding elements to Pipeline \n")
    pipeline.add(pgie)
    pipeline.add(tracker)
    pipeline.add(nvdsanalytics)
    pipeline.add(tiler)
    pipeline.add(nvvidconv)
    pipeline.add(filter1)
    pipeline.add(nvvidconv1)
    pipeline.add(nvvmsgconv)
    pipeline.add(nvosd)
    if is_aarch64():
        pipeline.add(transform)
    pipeline.add(sink)
    print("Linking elements in the Pipeline \n")
    streammux.link(pgie)
    pgie.link(tracker)
    tracker.link(nvdsanalytics)
    nvdsanalytics.link(nvvidconv1)
```

```
nvvidconv1.link(filter1)
filter1.link(tiler)
tiler.link(nvvmsgconv)
nvvmsgconv.link(nvvidconv)
nvvidconv.link(nvosd)
if is_aarch64():
    nvosd.link(transform)
    transform.link(sink)
else:
    nvosd.link(sink)
```

Based on the program above, there are two data linked to the destination element or sink. First, the metadata with RGBA buffer data type and metadata contains the JSON data type. The first metadata will display the result on the screen, including object bounding boxes and line crossing analytics. The second metadata contains information related to the analytics, such as the total number of detected vehicles and the vehicle's counting crossed specific line crossing.

Figure 4.2: AddInsight-Vision Initial Result (Daylight Situation)

Figure 4.3: AddInsight-Vision Initial Result (Night Situation)

Based on Figures 4.2 and 4.3, the performance of proposed intelligent video system, AddInsight-Vision, must be tested and evaluated. In this project, the system evaluation is divided into two processes, model testing, and model evaluation. Model testing requires explicit behavior checks, which should be followed by a designed system. The model evaluation covers quantitative metrics and plots based on a specific baseline to summarise performance on a test or validation dataset.

In this project, a set of checklists is created to perform a qualitative analysis

of AddInsight vision.

Item	Yes/No
The AddInsight-Vision works without showing error message	Yes
The objects including car, person, bus, and other vehicles is detected.	No
The AddInsight-Vision able to send the data to AddInsight-Dashboard	Yes

Table 4.1: AddInsight-Vision Model Testing Check-List

The checklist showed that the AddInsight was not able to detect several objects of traffic scenes. Therefore, a transfer learning process was conducted in order to increase the performance of the AddInsight-Vision streaming pipeline.

4.2 Object Detection Transfer Learning

To increase the accuracy of the object detection model of AddInsight-Vision, several transfer learning processes are conducted on data based on the DPTI dataset. The transfer learning is conducted using the Tensorflow framework installed on a Desktop PC with the support of 16GB of RAM, Geforce GTX 1650 GPU, and Intel Core 17 CPU. The first step of transfer learning is dataset preparation. In this project, 7180 images and labels were created from 6 videos of traffic scenes situation between 08.00 AM to 07.00 PM with the details described in Table 4.2.

Data	Total Image	Total Label
East Camera – 08.00	1147	1147
West Camera – 08.00	1147	1147
East Camera – 19.00	1511	1511
West Camera – 19.00	3375	3375
Total	7180	7180

Table 4.2: DPTI Dataset.

The dataset is picked due to the aim of this project to observe the performance of AddInsight-Vision to detect objects under specific conditions, such as object occlusion, light condition, and camera position. From each of the datasets, the

data were randomly selected and partitioned into 70% training data and 30% evaluation data. Based on this dataset, the total ground truth label for each object is described in Table 4.3.

In this project, strategy two is chosen as the transfer learning method, which means only the classification layer is modified to meet with the target class. Based on the dataset, there six target classes that should be trained. Hence, the extended class of the network classifier would be

```
 target_class_mapping {
    key: "person"
    value: "person"
}
target_class_mapping {
    key: "car"
    value: "car"
}
  target_class_mapping {
    key: "bus"
    value: "bus"
}
  target_class_mapping {
    key: "truck"
    value: "truck"
}

  target_class_mapping {
    key: "motorcycle"
    value: "motorcycle"
}
  target_class_mapping {
    key: "bicycle"
    value: "bicycle"
}
```

The input data of the CNN models are based on the 3-channel image with 480 x 288 dimension and the augmentation process including, image pre-processing, spatial augmentation, and colour augmentation. The transfer learning

configuration is described in Table 4.4 .

Furthermore, four different CNN models, including DetectNet, SSD, RetinaNet, and Faster R-CNN, is trained with extended target class in order to increase the accuracy of object detection. The evaluation result will be shown in the following section.

4.3 Vehicle Re-Identification Transfer Learning

In this project, a vehicle re-identification model is proposed to achieve one of the aims of an intelligent transport system which is object tracking from multiple cameras. To achieve the aim, using an improved joint loss function, the ResNet-50 convolutional neural network is re-trained to determine the performance of vehicle re-identification model based on VeRi dataset. The transfer learning is conducted using Pytorch framework installed on Desktop PC with the support of 16GB of RAM, Geforce GTX 1650 GPU, and Intel Core 17 CPU. Furthermore, the transfer learning configuration is described in Table 4.5.

Data	Bicycle	Bus	Car	Motorcycle	Person	Truck
East Camera – 08.00	14	1404	1854	11	156	23
West Camera – 08.00	40	1230	3171	16	995	16
East Camera – 19.00	34	460	1619	13	113	5
West Camera – 19.00	28	1193	6980	122	3002	0

Table 4.3: DPTI Dataset Total Ground-truth Label.

Configuration	**Value**
Number of Epochs	120
Batch Size per GPU	8
Learning Rate	Max: 1.5e-2, Min: 4e-5

Table 4.4: Object Detection Transfer Learning Configuration

Configuration	Value
Number of Epochs	60
Batch Size per GPU	8
Learning Rate	0.0004
Training Data	37,778 Images, 37,778 Labels
Testing Data	11,579 Images, 11,579 Labels
Query Data	1,678 Images, 1,678 Labels

Table 4.5: Vehicle Re-ID Transfer Learning Configuration

4.4 System Evaluation

In this project, quantitative analysis using several metrics are used to determine the performance of object detection and vehicle re-identification model. The evaluation metrics of object detection model is Mean Average Precision (MAP). In order MAP to evaluate the detector, one should understand the concept of Intersection Over Union (IoU). IoU measures the overlap between two boundaries, given the ground truth boundaries described in the target label and predicted boundaries as a result of object detection (Figure 4.4). To state whether the object is truly positive or false positive, one should determine the threshold of IoU. In this project, the threshold IOU is based on the baseline of VOC2011 challenge, which is 0.5.

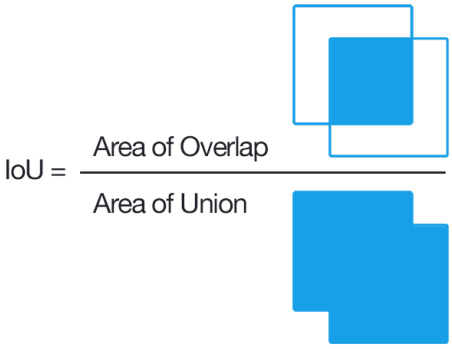

Figure 4.4: IoU Illustration ("Intersection over Union (IoU) for object detection - PyImageSearch," 2020)

Using the confusion matrix, the result of object detection is determined based on its IoU values. True Positive(TP), if $IoU > 0.5$ and match target class. False Positive (FN), if $IoU < 0.5$. False Negative (FN) if if $IoU > 0.5$ and match wrong class. True Negative (TN) if there is no detection. Given the confusion matrix, precision and recall function are determined.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Precision measures the accuracy of prediction, whereas recall measure how many positives occurred in the evaluation process. Using precision p and recall r curve, one can calculate average precision using the function below.

$$AP = \int_0^1 p(r)dr \tag{4.1}$$

In addition, Mean Average Precision calculates the average AP over the targeted class in the object detection. Furthermore, the results of transfer learning based on four datasets shown in the figures below.

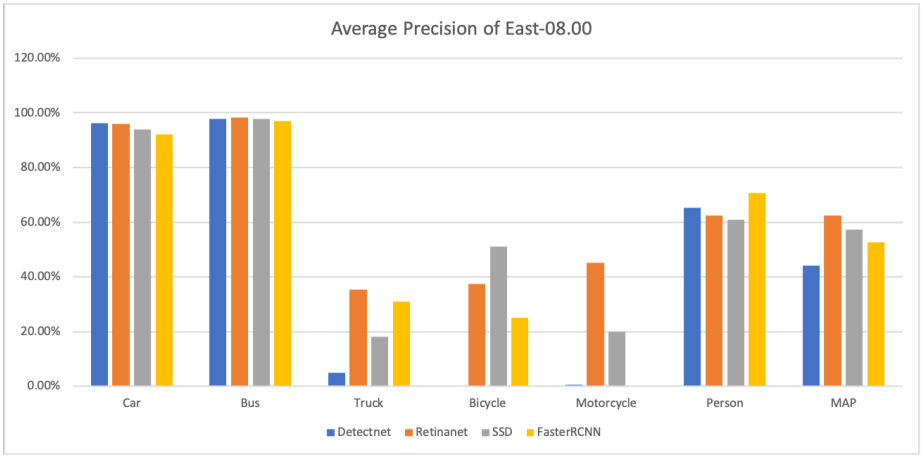

Figure 4.5: MaP Result of East-08.00 Data

Based on the figures (Figure 4.5 - Figure 4.8), they show that transfer learning increases the accuracy of the detection, which for several classes, the accuracy

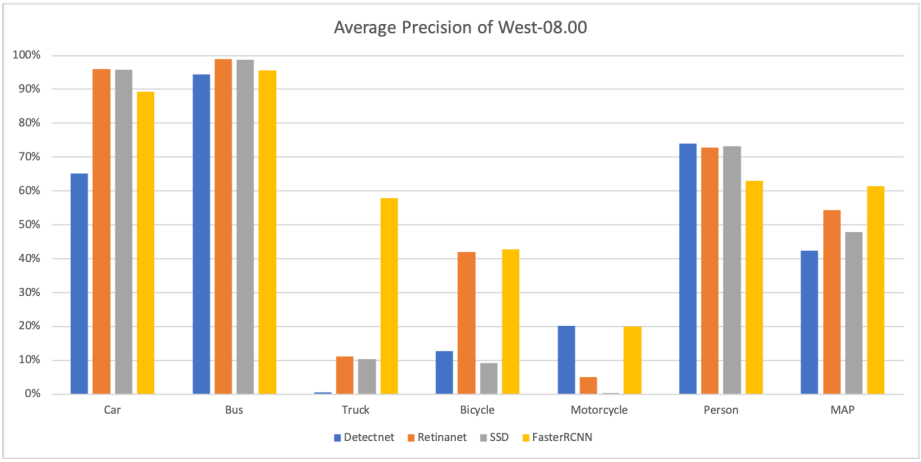

Figure 4.6: MaP Result of West-08.00 Data

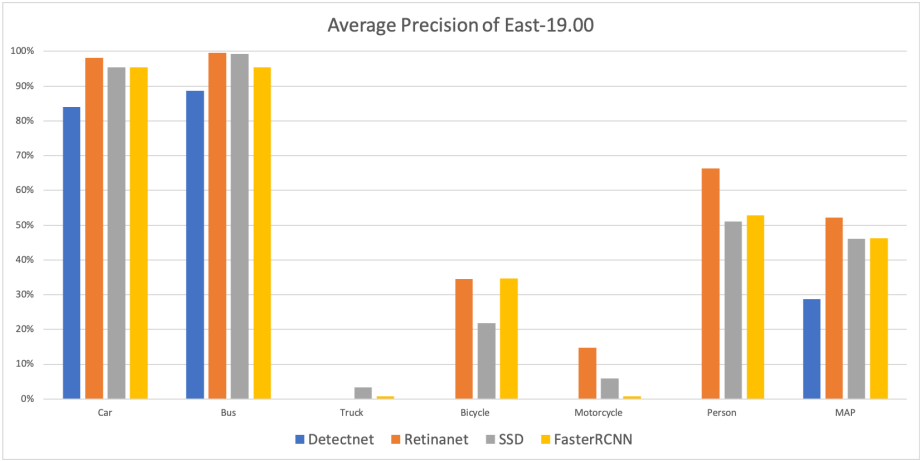

Figure 4.7: MaP Result of East-19.00 Data

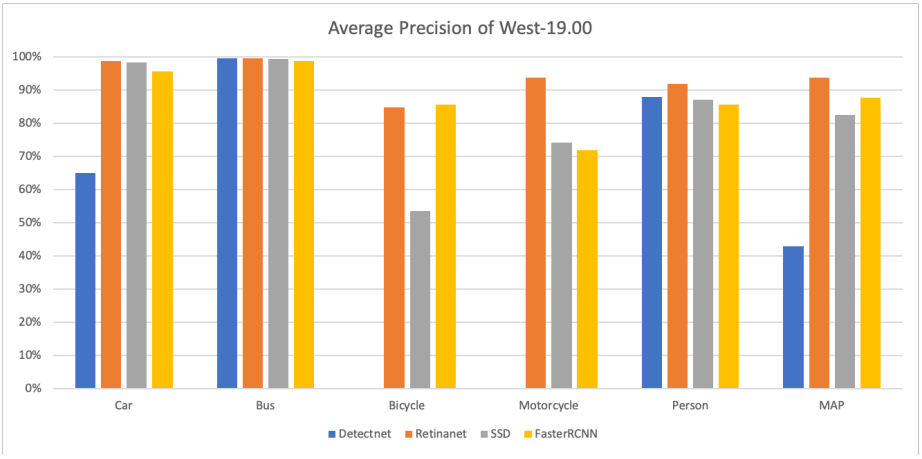

Figure 4.8: MaP Result of West-19.00 Data

achieved over 90%. However, due to the limitation ground truth labels on a class such as truck, bicycle, and motorcycle, the accuracy did not improve significantly. The figures also show that from 3 datasets, RetinaNet outperformed other three detectors. Furthermore, by using transfer learning, the performance of object detection models of AddInsight-Vision could be increased. In Figure 4.6 it is shown that FasterRCNN model outperformed other models since in the particular dataset, truck class has more number than other dataset.

Figure 4.9: Result of AddInsight-Vision's Object Detection Model after Transfer Learning

The second metric used for quantitative analysis of AddInsight-Vision is Cumulative matching characteristic (CMC). CMC is used to evaluate the performance of Vehicle Re-Identification model. CMC will rank all the samples in the testing directory based on their distance similarity with samples from query dataset. The top k CMC accuracy follows this equation (Nandakumar

et.al, 2009).

$$Acc_k = \begin{cases} 1 & \text{if top-}k\text{ ranked gallery samples contain the query identity} \\ 0 & \text{otherwise} \end{cases},$$

This formula resulted in a shifted step function. The CMC curve is calculated by averaging the shifted step function for all queries. The CMC curve of the proposed vehicle re-identification model is shown in Figure 4.10. The comparison is made to the other vehicle re-identification baseline.

Image removed due to copyright restriction.

Figure 4.10: CMC Results Compared with Another Baseline (Li and Zhou, 2019)

From the figure, it is shown that the vehicle re-identification could achieve second-best of the vehicle re-identification baseline. Furthermore, by using joint loss function for vehicle re-identification model, it could achieve a well-performed result and outperformed another baseline that only uses a single loss function.

In this project, the end-to-end performance of the AddInsight-Pipeline also measured by the Frame Rate Performance of each implemented object detection models. Table x shows a comparison of FPS for four different detectors, DetectNet, RetinaNet, SSD, Faster R-CNN.

Item	DetecNet	RetinaNet	SSD	Faster R-CNN
FPS	25	24.8	24	18-19
Power	30 W	30 W	30 W	30 W

Table 4.6: FPS Result of Four Different Object Detectors

Based on the FPS result, DetectNet outperformed other detectors. Specifically for Faster R-CNN, it suffers the lowest FPS result since Faster R-CNN is a two-stage detector which requires a longer time to process the detection.

4.5 AddInsight-Vision Edge Infrastructure Deployment

AddInsight-Vision will be deployed to the city of Launceston in Tasmania, Australia. The current progress of AddInsight-Vision deployment plan is the SAGE Automation team still prepare the remote-control feature to allow the engineers to configure and monitor the performance of AddInsight-Vision. This deployment aims to test AddInsight-Vision pipeline with the real-time situation and projected to be finished before the end of 2020.

Chapter 5

Conclusion and Future Works

5.1 Research Findings and Outcomes

In this project, an end-to-end intelligent video system called AddInsight-Vision is proposed as an implementation of the Intelligent Traffic System. AddInsight-Vision is able to detect, track, and monitor object in the traffic scenes situation, and gather the traffic information precisely. The AddInsight-Vision also integrated with current AddInsight system where the traffic information is able to be transmitted to AddInsight dashboard. The following results are obtained based on the development of AddInsight-Vision:

1. A framework of intelligent video system development is presented. The framework covers the process from determining the use-case of the project to the deployment stage. The aim of this framework is to create effective and efficient intelligent video system.

2. AddInsight-Vision, an intelligent video system based on stream processing paradigm, is developed. AddInsight-Vision is the state-of-art of stream processing that able to detect objects in traffic scenes with high accuracy. Using RetinaNet as the base of object detection, the accuracy of AddInsight-Vision to detect car and bus achieved 90% while maintaining well-performed operation. AddInsight-Vision is also able to be deployed on edge, to allow stakeholders to monitor traffic situation without the needs of high-end hardware infrastructure.

3. A DPTI dataset is developed based on the traffic scenes video from the South Australian Department of Planning, Transport, and Infrastructure.

The dataset consists of 7180 images and 7180 labels that were used for transfer learning and evaluation purposes.

4. An improved vehicle re-identification model is developed and outperformed other models on the re-identification baseline. AddInsight-Vision vehicle re-identification model is developed by implementing transfer learning on Resnet-50 convolutional neural network. The proposed joint loss function achieved 92% in the rank-5 accuracy of the cumulative matching characteristic metric.

5.2 Study Limitation

1. Dataset

   One of the limitations in the development of computer vision and deep learning project is to choose a suitable dataset. In this project, only two datasets were used, DPTI dataset and Ve-RI dataset. The datasets contain less ground truth label for several class such as trucks, bicycle and motorcycle, than other class. Thus the resulted transfer learning of these classes suffered low accuracy.

2. The stream processing pipeline

   The AddInsight-Vision is developed using Deepstream SDK from NVIDIA. The SDK might not be able to be implemented in the embedded device from other manufacturers.

3. Source Code

   Due to the nature of this project is an industrial project, AddInsight-Vision source code is not publicly accessible.

5.3 Future Works

1. Integration of Vehicle Re-Id

   The future work for AddInsight-Vision is to integrate vehicle re-id model

to the pipeline. The integration must consider the dataset since the vehicle re-id models specifically designed for multi-camera integration.

2. On-the-Fly System

   The deployment of AddInsight-System to the edge requires real-time transfer learning and evaluation. Based on this condition, the future work of AddInsight-Vision will focus on enabling real-time transfer learning without using additional hardware.

3. The ethical concern of Intelligent Video System Surveillance of large traffic system and the large crowd must consider the ethical concern of targeted object. One of the suggestions is to create a filter to blur the face of person or pedestrian.

Bibliography

[1] Adeleke, O.O., Idoko, S., Kolo, S.S., Anwar, A.R., Sijuwola, O.O., Akinola, O., 2019. Web-Based Advanced Traveller Information System for Minna Metropolis, Nigeria. 1 15, 1026–1037.

[2] Anand, B., Barsaiyan, V., Senapati, M., Rajalakshmi, P., 2020. Region of Interest and Car Detection using LiDAR data for Advanced Traffic Management System, in: 2020 IEEE 6th World Forum on Internet of Things (WF-IoT). Presented at the 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), pp. 1–5. https://doi.org/10.1109/WF-IoT48130.2020.9221354

[3] Barbagli, B., Manes, G., Facchini, R., Manes, A., 2012. Acoustic Sensor Network for Vehicle Traffic Monitoring 6.

[4] Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934 [cs, eess].

[5] Chen, J., Ran, X., 2019. Deep Learning With Edge Computing: A Review. Proceedings of the IEEE PP, 1–20. https://doi.org/10.1109/JPROC.2019.2921977

[6] Dong, J., Wang, F., Yuan, B., 2013. Accelerating BIRCH for Clustering Large Scale Streaming Data Using CUDA Dynamic Parallelism, in: Yin, H., Tang, K.,

[7] Gao, Y., Klawonn, F., Lee, M., Weise, T., Li, B., Yao, X. (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2013, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 409–416. https://doi.org/10.1007/978-3-642-41278-3_50

[8] Elliott, D., 2010. Intelligent video solution: A definition. Security 47.

[9] Fang, Y., Sun, L., Fu, H., Wu, T., Wang, R., Dai, B., 2016. Learning deep compact channel features for object detection in traffic scenes, in: 2016 IEEE International Conference on Image Processing (ICIP). Presented at the 2016 IEEE International Conference on Image Processing (ICIP), pp. 1052–1056. https://doi.org/10.1109/ICIP.2016.7532518

[10] Freund, Y., Schapire, R.E., 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55, 119–139. https://doi.org/10.1006/jcss.1997.1504

[11] Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research 32, 1231–1237. https://doi.org/10.1177/0278364913491297

[12] Geng, M., Wang, Y., Xiang, T., Tian, Y., 2016. Deep Transfer Learning for Person Re-identification. arXiv:1611.05244 [cs].

[13] Ghavami, S.M., 2020. A web service based advanced traveller information system for itinerary planning in an uncertain multimodal network. Geocarto International 35, 1553–1569. https://doi.org/10.1080/10106049.2019.1583773

[14] Girshick, R., 2015. Fast R-CNN. arXiv:1504.08083 [cs].

[15] Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524 [cs].

[16] Guerrero-Ibañez, J., Zeadally, S., Contreras Castillo, J., 2018. Sensor Technologies for Intelligent Transportation Systems. Sensors 18, 1212. https://doi.org/10.3390/s18041212

[17] Hermans, A., Beyer, L., Leibe, B., 2017. In Defense of the Triplet Loss for Person Re-Identification. arXiv:1703.07737 [cs].

[18] Ibrahim, M.R., Haworth, J., Cheng, T., 2020. Understanding cities with machine eyes: A review of deep computer vision in urban analytics. Cities 96, 102481. https://doi.org/10.1016/j.cities.2019.102481

[19] Intersection over Union (IoU) for object detection - PyImageSearch [WWW Document], n.d. URL https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (accessed 11.16.20).

[20] Jeong, H., Choi, S., Jang, S., Ha, Y., 2019. Driving Scene Understanding Using Hybrid Deep Neural Network, in: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). Presented at the 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 1–4. https://doi.org/10.1109/BIGCOMP.2019.8679323

[21] Kim, K.G., 2016. Book Review: Deep Learning. Healthc Inform Res 22, 351. https://doi.org/10.4258/hir.2016.22.4.351

[22] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 84–90. https://doi.org/10.1145/3065386

[23] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V., 2020. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. Int J Comput Vis 128, 1956–1981. https://doi.org/10.1007/s11263-020-01316-z

[24] Lai, L., Suda, N., 2018. Enabling deep learning at the IoT edge, in: Proceedings of the International Conference on Computer-Aided Design, ICCAD '18. Association for Computing Machinery, New York, NY, USA, pp. 1–6. https://doi.org/10.1145/3240765.3243473

[25] Lee, K., Yim, K., Mikki, M.A., 2012. A secure framework of the surveillance video network integrating heterogeneous video formats

and protocols. Computers and Mathematics with Applications, Advances in context, cognitive, and secure computing 63, 525–535. https://doi.org/10.1016/j.camwa.2011.08.048

[26] Li, X., Zhou, Z., 2019. Object Re-Identification Based on Deep Learning. Visual Object Tracking with Deep Neural Networks. https://doi.org/10.5772/intechopen.86564

[27] Li, Y., Li, Y., Yan, H., Liu, J., 2017. Deep joint discriminative learning for vehicle re-identification and retrieval, in: 2017 IEEE International Conference on Image Processing (ICIP). Presented at the 2017 IEEE International Conference on Image Processing (ICIP), pp. 395–399. https://doi.org/10.1109/ICIP.2017.8296310

[28] Lienhart, R., Maydt, J., 2002. An extended set of Haar-like features for rapid object detection, in: Proceedings. International Conference on Image Processing. Presented at the Proceedings. International Conference on Image Processing, p. I–I. https://doi.org/10.1109/ICIP.2002.1038171

[29] Lin, C., Tang, Y., 2011. Research and design of the intelligent surveillance system based on DirectShow and OpenCV, in: 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet). Presented at the 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet), pp. 4307–4310. https://doi.org/10.1109/CECNET.2011.5768334

[30] Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2018. Focal Loss for Dense Object Detection. arXiv:1708.02002 [cs].

[31] Lin, Y., Zheng, L., Zheng, Z., Wu, Y., Hu, Z., Yan, C., Yang, Y., 2019. Improving Person Re-identification by Attribute and Identity Learning. Pattern Recognition 95, 151–161. https://doi.org/10.1016/j.patcog.2019.06.006

[32] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. arXiv:1512.02325 [cs]. https://doi.org/10.1007/978-3-319-46448-0_2

[33] Liu, X., Liu, W., Ma, H., Fu, H., 2016. Large-scale vehicle re-identification in urban surveillance videos, in: 2016 IEEE International Conference on Multimedia and Expo (ICME). Presented at the 2016 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6. https://doi.org/10.1109/ICME.2016.7553002

[34] Lowe, D.G., 2004. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[35] MARIED, E., ELDALI, M., ZIADA, O., Baba, A., 2017. A Literature Study of Deep learning and its application in Digital Image Processing. https://doi.org/10.13140/RG.2.2.17403.72480

[36] McGowen, P., Sanderson, M., 2011. Accuracy of Pneumatic Road Tube Counters. Presented at the Institute of Transportation Engineers (ITE). Western District Annual Meeting, 2011.

[37] Nikouei, S.Y., Chen, Y., Song, S., Xu, R., Choi, B., Faughnan, T.R., 2018. Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN, in: 2018 IEEE International Conference on Edge Computing (EDGE). Presented at the 2018 IEEE International Conference on Edge Computing (EDGE), pp. 125–129. https://doi.org/10.1109/EDGE.2018.00025

[38] NVIDIA, 2020. DeepStream SDK 5.0. for NVIDIA dGPU and Jetson 13.

[39] NVIDIA, 2016. DetectNet: Deep Neural Network for Object Detection in DIGITS [WWW Document]. NVIDIA Developer Blog. URL https://developer.nvidia.com/blog/detectnet-deep-neural-network-object-detection-digits/ (accessed 11.10.20).

[40] Peng, J., Nan, Z., Xu, L., Xin, J., Zheng, N., 2020. A Deep Model for Joint Object Detection and Semantic Segmentation in Traffic Scenes, in: 2020 International Joint Conference on Neural Networks (IJCNN). Presented at the 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9206883

[41] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640 [cs].

[42] Ren, S., He, K., Girshick, R., Sun, J., 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497 [cs].

[43] Ren, X., Wang, D., Laskey, M., Goldberg, K., 2018. Learning Traffic Behaviors by Extracting Vehicle Trajectories from Online Video Streams, in: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). Presented at the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), pp. 1276–1283. https://doi.org/10.1109/COASE.2018.8560597

[44] Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N., 2009. The Case for VM-Based Cloudlets in Mobile Computing. IEEE Pervasive Computing 8, 14–23. https://doi.org/10.1109/MPRV.2009.82

[45] Simonyan, K., Zisserman, A., 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs].

[46] Singh, B., Gupta, A., 2015. Recent trends in intelligent transportation systems: a review. J. Transp. Lit. 9, 30–34. https://doi.org/10.1590/2238-1031.jtl.v9n2a6

[47] Stephens, R., 1997. A survey of stream processing. Acta Informatica 34, 491–541. https://doi.org/10.1007/s002360050095

[48] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2015. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567 [cs].

[49] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A Survey on Deep Transfer Learning. arXiv:1808.01974 [cs, stat].

[50] Taylor, B., Marco, V.S., Wolff, W., Elkhatib, Y., Wang, Z., 2018. Adaptive deep learning model selection on embedded systems, in: Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems - LCTES 2018. Presented at the the 19th ACM SIGPLAN/SIGBED International Conference, ACM Press, Philadelphia, PA, USA, pp. 31–43. https://doi.org/10.1145/3211332.3211336

[51] Watcharapinchai, N., Rujikietgumjorn, S., 2017. Approximate license plate string matching for vehicle re-identification, in: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). Presented at the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6. https://doi.org/10.1109/AVSS.2017.8078538

[52] Ye, T., Zhang, Z., Zhang, X., Zhou, F., 2020. Autonomous Railway Traffic Object Detection Using Feature-Enhanced Single-Shot Detector. IEEE Access 8, 145182–145193. https://doi.org/10.1109/ACCESS.2020.3015251

[53] You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J., Keutzer, K., 2018. ImageNet Training in Minutes. arXiv:1709.05011 [cs].

[54] Zhang, X., 2017. Support Vector Machines, in: Sammut, C., Webb, G.I. (Eds.), Encyclopedia of Machine Learning and Data Mining. Springer US, Boston, MA, pp. 1214–1220. https://doi.org/10.1007/978-1-4899-7687-1_810

[55] Zhang, Y., Liu, D., Zha, Z., 2017. Improving triplet-wise training of convolutional neural network for vehicle re-identification, in: 2017 IEEE International Conference on Multimedia and Expo (ICME). Presented at the 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 1386–1391. https://doi.org/10.1109/ICME.2017.8019491

[56] Zhou, Y., Chen, Z., Huang, X., 2015. A pipeline architecture for traffic sign classification on an FPGA, in: 2015 IEEE International Symposium on Circuits and Systems (ISCAS). Presented at the 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 950–953. https://doi.org/10.1109/ISCAS.2015.7168792