





Article

FGSC: Fuzzy Guided Scale Choice SSD Model for Edge AI Design on Real-Time Vehicle Detection and Class Counting

Ming-Hwa Sheu ¹, S. M. Salahuddin Morsalin ^{1,*} , Jia-Xiang Zheng ¹, Shih-Chang Hsia ¹ , Cheng-Jian Lin ²  and Chuan-Yu Chang ³ 

¹ Department of Electronic Engineering, National Yunlin University of Science and Technology, Douliu 64002, Taiwan; sheumh@yuntech.edu.tw (M.-H.S.); jamesb09111@gmail.com (J.-X.Z.); hsia@yuntech.edu.tw (S.-C.H.)

² Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411030, Taiwan; cjlin@ncut.edu.tw

³ Computer Science and Information Engineering, National Yunlin University of Science and Technology, Douliu 64002, Taiwan; chuanyu@yuntech.edu.tw

* Correspondence: s.morsalin10@gmail.com

Abstract: The aim of this paper is to distinguish the vehicle detection and count the class number in each classification from the inputs. We proposed the use of Fuzzy Guided Scale Choice (FGSC)-based SSD deep neural network architecture for vehicle detection and class counting with parameter optimization. The ‘FGSC’ blocks are integrated into the convolutional layers of the model, which emphasize essential features while ignoring less important ones that are not significant for the operation. We created the passing detection lines and class counting windows and connected them with the proposed FGSC-SSD deep neural network model. The ‘FGSC’ blocks in the convolution layer emphasize essential features and find out unnecessary features by using the scale choice method at the training stage and eliminate that significant speedup of the model. In addition, FGSC blocks avoided many unusable parameters in the saturation interval and improved the performance efficiency. In addition, the Fuzzy Sigmoid Function (FSF) increases the activation interval through fuzzy logic. While performing operations, the FGSC-SSD model reduces the computational complexity of convolutional layers and their parameters. As a result, the model tested Frames Per Second (FPS) on edge artificial intelligence (AI) and reached a real-time processing speed of 38.4 and an accuracy rate of more than 94%. Therefore, this work might be considered an improvement to the traffic monitoring approach by using edge AI applications.

Keywords: fuzzy guided scale choice; fuzzy sigmoid function; vehicle detection; fuzzy logic; vehicle class counting; and intelligent AIoT vehicles application



Citation: Sheu, M.-H.; Morsalin, S.M.S.; Zheng, J.-X.; Hsia, S.-C.; Lin, C.-J.; Chang, C.-Y. FGSC: Fuzzy Guided Scale Choice SSD Model for Edge AI Design on Real-Time Vehicle Detection and Class Counting. *Sensors* **2021**, *21*, 7399. <https://doi.org/10.3390/s21217399>

Academic Editor: Roberto Teti

Received: 13 September 2021

Accepted: 3 November 2021

Published: 7 November 2021

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Vehicle detection, classification, counting, and monitoring all are regular tasks for the traffic management authority. The volume of vehicles on the roads has an economic impact on the transportation sector and the specific region around the country [1]. Geomagnetic simulation, radio-frequency detection, or vision-based technology are all used to monitor vehicles. Geomagnetic simulation [2] needs to be built beneath the road’s surface, and its execution is quite complicated and expensive. The radio-frequency detection method [3,4] has a good effect on results. In addition, the radio-frequency detection method is suitable for vehicle counting and highway toll ticketing systems in favor of the e-Tag. However, the approach requires an elevated bridge, which is not always accessible everywhere, making it costly and inconvenient for the vendors. However, the image detection approach [5–7] is simple to set up and maintain in any location. In addition, the image detection method also can be directly combined with road surveillance cameras. Since there are many security

cameras on highways, roads, and streets nowadays, image detection can quickly be related to the existing system.

Computer vision systems extract the salient regions from the background of an input image [8]. Many existing CNN-based salient object identification techniques aim to learn feature vectors at various scales of picture portions, inferring the saliency of each part in the image [9]. The necessity of many labeled images as data in the training stage is a fundamental challenge in the existing deep saliency detection algorithms. The training processes for pixel-wise ground-truth annotation techniques, on the other hand, are time-consuming [10]. Salient Object Detection (SOD) methods expand the object detection, segmentation, and annotating of the training data [11] without human efforts. Although SOD methods provide saliency maps of object areas as output, detecting tiny objects and achieving exact location remains problematic [12]. The SSD [13] model improves default box output space, varied aspect ratios, feature map placement, and the modification of default and bounding boxes to better fit object appearances. SSD has achieved significant progress in detecting small-sized objects and increased localization accuracy with multiscale feature maps and the default boxes method. We adopted the SSD model and extended it with the fuzzy guided scale choice block.

Block sparse RSPCA [5] is useful for vehicle detection and counting. Using digital signal processing-based image recognition technology, the total number of vehicles on the road can be counted. The authors of [6] proposed vehicle counting with real-time proceeding speeds, but their approach used the background subtraction method of low-rank sparse to detect foreground moving objects and counting vehicles. We have considered an AI-based method for vehicle object detection and class counting. Fast-SSD [7] is offered for vehicle identification and counting, and it recognizes objects using six-scale feature maps, resulting in good accuracy. However, Fast-SSD can operate only on one-way roads. Furthermore, the SSD model is a sequential convolution approach, which causes many weights to become zero, wasting time and resulting in low efficiency [14]. As a result, we proposed the FGSC-SSD model for forward passing vehicle detection and backward passing vehicle detection, separately, and class counting. The FGSC-SSD model solves the efficient loss of multiple convolutional layers by utilizing learning weights to impose important and irrelevant features. The operation choice mechanism skips the unimportant characteristics that would have robust effects on the operation speed during the test stage. The main contributions of this work are summarized as follows:

- ❖ We set up the Region of Interest (ROI) for quick vehicle detection and class counting from forward passing and backward passing lanes.
- ❖ Proposed FGSC blocks distinguish between significant and irrelevant features. Next, improved system operation speed by skipping unnecessary characteristics.
- ❖ We have developed a fuzzy sigmoid function for controlling the activation interval and avoiding saturated output feature values.
- ❖ In comparison to the SSD model, the proposed FGSC-SSD model has achieved higher speed under the same detection accuracy for the PASCAL VOC dataset and Benchmark BIT Vehicle dataset.

The FGSC-SSD model-based vehicle detection and class counting system has been deployed on edge AI and in real-time. Its processed speed is about 38.4 FPS for real video. The total accuracy rates achieved for cars, buses, and trucks was 95.1%, 92.3%, and 90.9%, accordingly.

2. Related Works

Object detection is more challenging for visual attention, drawing the bounding box around each object of interest and assigning them a class. The deep learning (DL) algorithm has made tremendous progress in object identification, demonstrating, classification, higher feature extraction, and considerably improving detection. One of the most successful applications is the convolutional neural network (CNN) AlexNet [15], which outperformed prior methods.

In the research of object tracking, the two-stage object detection deep neural network models are SPPNet [16], R-CNN [17], Fast R-CNN [18], Faster R-CNN [19], Mask R-CNN [20], ME R-CNN [21], MFR-CNN [22], SWAE [23], and A CoupleNet [24], which all improved performance. The models of the two stages need a Region Proposal Network (RPN) to extract bounding boxes. Those bounding boxes are like ground-truth objects which can precisely locate the object position with good performance. While two-stage solves the instance segmentation problem, it also adds to the network's processing cost and computational complexity.

For considering the speed performance of the deep learning algorithm, some one-stage methods such as, for example, YOLO [25], SSD [26], Retina-Net [27], SqueezeDet [28], CornerNet [29], MSA-DNN [30], Image base [31], and DF-SSD [32], represent the object detection deep neural network models. Accuracy and speed are frequently incompatible, as can be seen when the accuracy of feature descriptors is significantly lower than that of a DL technique. The one-stage eliminates the RPN and ROI pooling, so the one-stage approach is faster than two-stage.

Table 1 depicts the performance of the object detection deep neural network models ME R-CNN, MFR-CNN, A-Couple-Net, and DF-SSD on the GPU platform, with lower FPS. The MFR-CNN projected to combine the multi-scale features and the global features improved the accuracy, but the lowest FPS was only 6.9. The DF-SSD tried to combine the shallow and deep characteristics to solve the SSD problem. The DF-SSD model achieved good accuracy for small objects, but the FPS was still low because of the lack of relation between shallow and deep features. The MSA-DNN proposed to combine MSA-DNN and MSAM and extract more features, which improved the accuracy and the FPS value, but those models cannot achieve real-time operation in edge AI. For the Real-Time application, the GPU performance of the SSD and YOLOv4 model's FPS is high than others for various vehicle detections, but it is hard to achieve real-time operation on an edge AI platform.

Table 1. Various Object Detection Model Performance.

Model	Platform	FPS	Real-Time
ME R-CNN	GPU	13.3	No
MFR-CNN	GPU	6.9	No
A-Couple-Net	GPU	9.5	No
DF-SSD	GPU	11.6	No
YOLOv4	Edge AI	13.6	No
SSD	Edge AI	16.9	No

The SSD model adds several feature layers at the end of a base network which predicts offsets in default boxes of different scales and aspect ratios and their associated confidences. The SSD model has multi-scale feature map technology for object detection, whereas the YOLO model has a single-scale feature map. In addition, the YOLO model has been developed by intermediate fully connected layers, whereas the SSD model employs coevolutionary filters for each feature map location. The SSD model detects objects using convolutional default boxes from multiple feature maps and matching strategies. In our study, we adopted the SSD model and extended this idea by applying the FGSC block architecture with the fuzzy sigmoid function to significantly increase the quality of the prediction and performance with an even smaller number of parameters.

3. Proposed Vehicle Detection and Class Counting System

The vehicle detection and class counting intelligent technique flow chart diagram in Figure 1 shows the working process in three phases. At first, the ROI setup can locate the position of the forward passing and backward passing objects through the object detection window from the road video as an input. The second step is the proposed FGSC-SSD deep neural network model to identify the vehicles from the current frame. The objects are categories as cars, buses, and trucks. Lastly, the vehicle class is counted based on

ROI windows setup and working principle. If the input video does not end, the process becomes repeated, following the vehicle detection and class counting flow chart. If it ends, the steps reach the end and show the result.

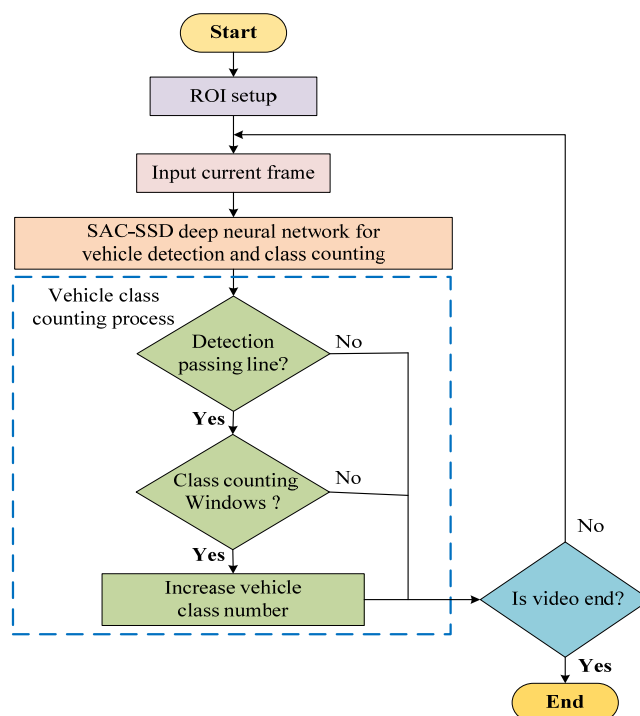


Figure 1. Vehicle detection and class counting flow chart.

3.1. ROI Setup

Figure 2 shows the ROI set up for the object detection window. The red and pink color frames represent the forward passing detection line and class counting window. Similarly, lime and dark green color frames are the backward passing detection line and the class counting window, correspondingly. Research [33] on automatic traffic counting systems showed the two-lane road has good performance on vehicle class counting.

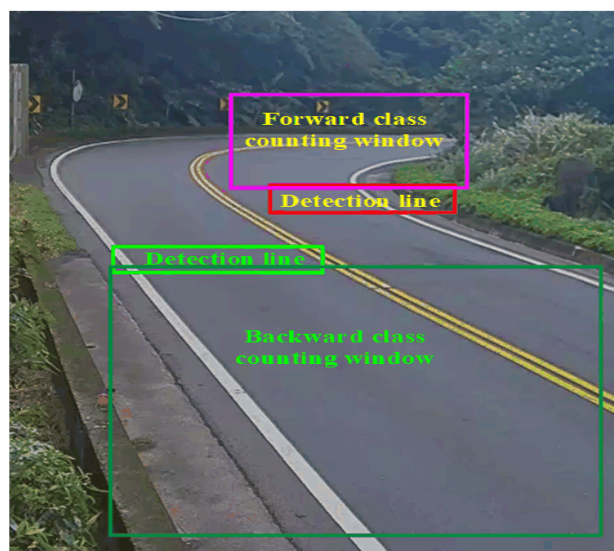


Figure 2. The vehicle's detection window.

The passing detection lines are responsible for checking whether the vehicles pass through on the detection lines or not. Consistently, the class counting windows are accountable for vehicle class counting. In both cases, an object passing and the class counting windows are placed behind the passing detection line.

3.2. FGSC-SSD Model Architecture

The continuous convolution operations of deep neural networks can approach the weight values to become zero, which causes many parameters to be invalid and inefficient for the operations. Therefore, the authors proposed the FGSC-SSD model architecture. Figure 3 demonstrated the proposed FGSC-SSD deep neural network model's architecture. The model belongs to eleven convolutional layers (Conv1 to Conv11), five FGSC blocks (FGSC1 to FGSC5), and six convolutional blocks (Conv12 to Conv17). Each layer and block consist of a different number of channels from input to output levels.

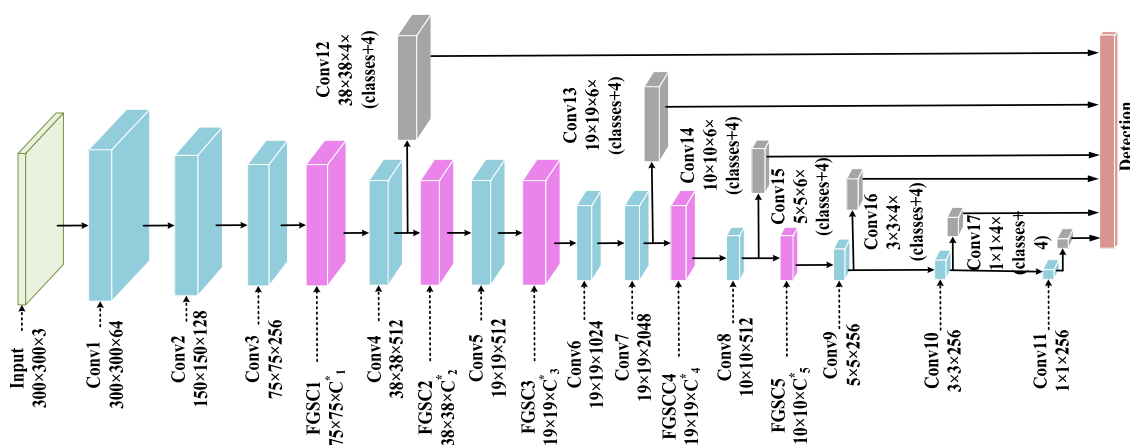


Figure 3. The proposed FGSC-SSD model architecture.

The five FGSC blocks have been incorporated into the convolutional layers of Conv3 to Conv9 in the FGSC-SSD model to reduce the performance loss by segmenting the essential characteristics. However, we did not integrate any FGSC blocks in between the input layer and Conv3 layer because shallow features belong to these layers. All initial features are significant for the experiment. Therefore, we cannot skip any characteristics during the initial operation for the first three convolutional layers. These FGSC blocks ignore the insignificant elements from the process and increase the detection speed at the test phase. Alternatively, the Conv9 to Conv11 layers are the deep convolutional layers and have fewer features that are important for the performance. Therefore, we have not added any FGSC blocks in the deep convolutional layers. The operational function of FGSC blocks depends on the global average pooling, fully connected, fuzzy sigmoid function, and scale choice layer. The design of the FGSC block enhances the convolutional operations and reduces the computational complexity.

We used another six convolutional blocks (Conv12–Conv17) for extracting the default boxes which predict the objects in the detection layer. Hard Negative Mining (HNM) was employed to control the positive and negative sample ratios at 3:1 and avoid the weights being biased toward negative samples with the features varying input sizes. The default box is used with the Ground Truth (GT) box to form the Intersection over Union (IoU). The IoU value for positive samples is less than 0.5, and the background probability is less than 0.3, indicating that the sample is a negative sample that can suppress the circumstance of the negative instances. The object location loss function combines convolutional network prediction and default frame prediction to calculate the object location loss of the GT frame corresponding to the positive sample. The combination of x and y coordinates, width, and height are shown in the results at the detection line.

3.3. FGSC Block Operation Function

The FGSC block architecture consists of the global intensity, fuzzy guided, and scale choice; the authors propose the global intensity function by the global average pooling [34], and the fuzzy guided function with the fully connected and fuzzy sigmoid. The fully connected layer combines global average pooling and fuzzy sigmoid function, which is shown in Figure 4. Therein, I denotes the block input and H , W , and C depict the input feature maps' height, width, and channel, respectively. The input feature maps I pass through the fuzzy guided scale functions. The fuzzy guided is fully connected with global intensity and the results of the fuzzy guided function, and I pass through the scale choice layer. The scale choice layer ensures optimized Y output selection for the blocks. The input channel number C changes into the C^* channel of the output feature map.

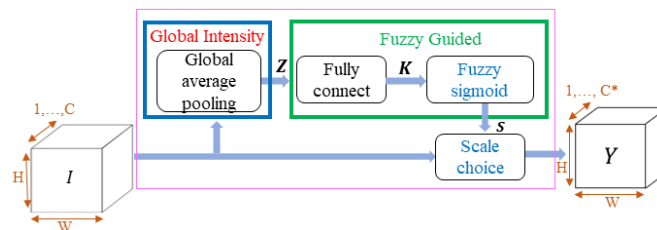


Figure 4. FGSC block architecture.

3.3.1. Global Intensity Function

The global average pooling extracts features for the global intensity function and is calculated through the following Equation (1). The global average pooling output features represent by $Z = \{z_1, z_2, \dots, z_c\}$. The FGSC block input feature $I = \{I_c(i, j), c = 1, \dots, C\}$ where C is the channel number, H is the height, and W is the width of I_c feature maps. The summation of input channel features I_c divided by the product of height and width values, then achieved as the result of the global average value.

$$z_c = \frac{1}{H \times W} \sum_i^H \sum_j^W I_c(i, j). \quad c = 1, 2, \dots, C \quad (1)$$

3.3.2. Fuzzy Guided

The fuzzy guide response to achieve the scale values is S . The fully connected and fuzzy sigmoid functions are proposed by the authors to obtain the output value of S . These scale values execute the important and unimportant features from block input I . The fuzzy sigmoid function depicts the fuzzy guide result, and we calculated the fuzzy guide scale value by the following Equation (2).

$$S = F_Sigmoid(fc(W_1, Z)) \quad (2)$$

- Fully Connected

Each channel of the global intensity and fuzzy sigmoid function combined by fully connected in the fuzzy guide operation for learning the average value of Z . The weights and biases of the fully connected Equation (3) are $W_1 \in R^{\frac{C}{r} \times C}$, r ; to control the memory size and calculation quantity, r is set to 2 in this study and $bias \in R^C$.

$$K = fc(W_1, Z) = Z * W_1 + bias \quad (3)$$

- Fuzzy Sigmoid Function

We incorporated the α parameter into the fuzzy sigmoid function that is shown in Equation (4), where $K = \{K_{min}, k_2, k_3 \dots K_{max}\}$ is the output value of the fully connect. The fully connected output's maximum and minimum values represent the α parameters.

The fuzzy range is defined by the highest and lowest fully connect output values. Each FGSC block achieved output values differently [35]. The fuzzy logic has better adaptability especially suitable for nonlinear function, so the authors announced the fuzzy logic to determine the α parameter of the fuzzy sigmoid function.

$$S = F_Sigmoid(K) = 1 / (1 + e^{-\alpha K}) \quad (4)$$

In addition, the fuzzy sigmoid function represents classification decisions explicitly in the form of fuzzy rules. This research has developed a new dimension of object detection with the support of the fuzzy sigmoid function. The fuzzy sigmoid function depicts the simple technology and the shortcomings to control the activation interval in the sigmoid function and uses fuzzy logic to select the best parameter.

In this part, we explained why to add α parameters into the fuzzy sigmoid function. The FGSC blocks training algorithm automatically deprives the fuzzy set parameters for the “fuzzy rules” in the fuzzy sigmoid function. The FGSC-SSD deep neural network-based object detection and class counting rules obtained, in symbolic form, facilitate the understanding. The maximum and minimum values are 20.539 and -20.310 for the FGSC block no. 2. Figure 5a shows the sigmoid function for $\alpha = 1$, and the upper and lower limits are 6 and -6 ; alternatively, Figure 5b represents the fuzzy sigmoid function set the α parameter to 0.09 so its upper and lower limits become 60 and -60 . Then the active area is between 20.539 and -20.310 will be in the saturation zone, so the gradient does not become ‘0’, and it can effectively update the weight values. The fuzzy sigmoid function uses the α value to control the activation interval and used fuzzy logic to find the appropriate α for effectively activating the parameters to improve performance.

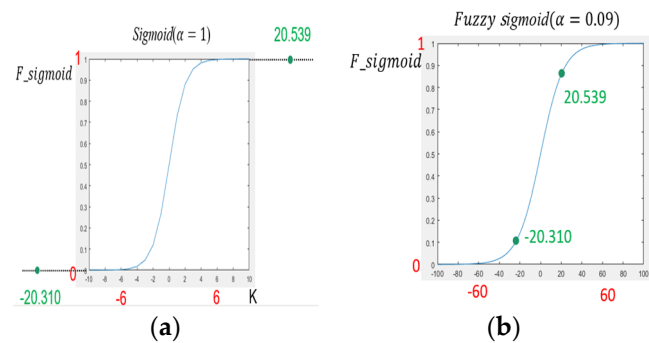


Figure 5. The function of (a) Sigmoid function. (b) Fuzzy sigmoid function.

1. The Fuzzy Sets for α

The $F_Sigmoid$ function has an asymmetric relationship between K_{max}^i and K_{min}^i values. To define the fuzzy sets, the maximum and minimum values of fuzzy sigmoid are obtained by fully connected output in the FGSC block. The K_{max}^i and K_{min}^i values are substitute into the following Equations (5)–(7) to get the non-isometric triangular fuzzy sets [36]. The ‘ $\tilde{A}_{\alpha_{mid}}^i$ ’, ‘ $\tilde{A}_{\alpha_{top}}^i$ ’, and ‘ $\tilde{A}_{\alpha_{bottom}}^i$ ’ are defined by the center, maximum, and minimum values which are obtained through a fuzzy set. After training the PASCAL VOC data set [37], the $F_Sigmoid$ function obtained 13 non-isometric triangular fuzzy sets $\tilde{A}_{\alpha} = \{\tilde{A}_{\alpha}^i, i = 1, \dots, 13\}$ that are shown in Figure 6.

$$\tilde{A}_{\alpha_{mid}}^i = \ln\left(\frac{1}{4}\right) / \left(\frac{|K_{max}^i| + |K_{min}^i|}{2}\right) \quad (5)$$

$$\tilde{A}_{\alpha_{top}}^i = \ln\left(\frac{1}{9}\right) / \left(\frac{|K_{max}^i| + |K_{min}^i|}{2}\right) \quad (6)$$

$$\tilde{A}_{\alpha_{bottom}}^i = \ln\left(\frac{3}{7}\right) / \left(\frac{|K_{max}^i| + |K_{min}^i|}{2}\right) \quad (7)$$

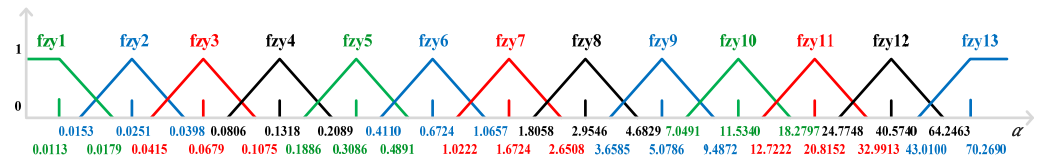


Figure 6. The \tilde{A}_{α} fuzzy set for α .

For example, after training the FGSC, block 1 has $K_{max}^{i=1} = 48.4701$, $K_{min}^{i=1} = -62.0113$. Those values are placed into Equations (5)–(7) and obtain the fuzzy set ‘fzy2’ mentioned in Figure 6, where $\tilde{A}_{\alpha_{mid}}^{i=1} = 0.0251$, $\tilde{A}_{\alpha_{top}}^{i=1} = 0.0398$ and $\tilde{A}_{\alpha_{bottom}}^{i=1} = 0.0153$.

2. The Fuzzy Sets for K_{max} and K_{min}

The following Figures 7 and 8 are the non-isometric triangular maximum and minimum fuzzy sets $\tilde{A}_{K_{max}} = \{\tilde{A}_{K_{max}}^i, i = 1, \dots, 13\}$ and $\tilde{A}_{K_{min}} = \{\tilde{A}_{K_{min}}^i, i = 1, \dots, 13\}$ for the K_{max} and, K_{min} values, respectively. According to the previous $\tilde{A}_{\alpha_{mid}}^i$ value, the authors attempted to construct the K_{max} and, K_{min} fuzzy sets based on the $F_Sigmoid$ function. The ‘ $\tilde{A}_{K_{max_mid}}^i$ ’ is the absolute average value for K_{max}^i and K_{min}^i that is achieved by the following Equation (8). The ‘ $\tilde{A}_{K_{max_top}}^i$ ’ is the top value of fuzzy set for ‘ $\tilde{A}_{K_{max}}^i$ ’ which is calculated by the Equation (9). The ‘ $\tilde{A}_{K_{max_bottom}}^i$ ’ is the bottom value for the fuzzy set of ‘ $\tilde{A}_{K_{max}}^i$ ’ that is calculated through Equation (10).

$$\tilde{A}_{K_{max_mid}}^i = \frac{|K_{max}^i| + |K_{min}^i|}{2} \quad (8)$$

$$\tilde{A}_{K_{max_top}}^i = -\ln\left(\frac{1}{9}\right) / \alpha_{mid}^i \quad (9)$$

$$\tilde{A}_{K_{max_bottom}}^i = -\ln\left(\frac{3}{7}\right) / \alpha_{mid}^i \quad (10)$$

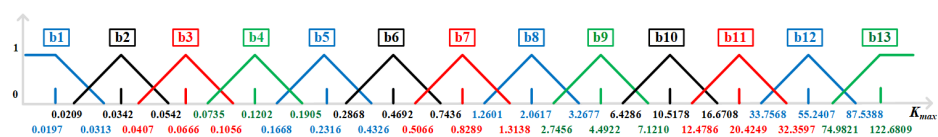


Figure 7. The $\tilde{A}_{K_{max}}$ maximum fuzzy set.

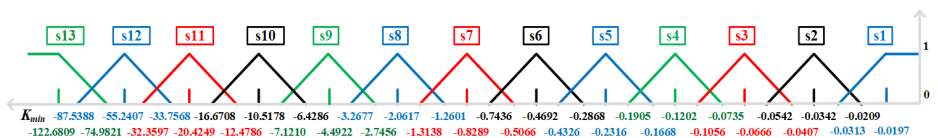


Figure 8. The $\tilde{A}_{K_{min}}$ minimum fuzzy set.

Similarly, in the FGSC block 1, the $K_{max}^{i=1}$ and $K_{min}^{i=1}$ values were substituted into Equations (8)–(10) and obtained a maximum fuzzy set ‘b12’ which is mentioned in Figure 7. The fuzzy set center value is ‘ $\tilde{A}_{K_{max_mid}}^i$ ’ = 55.2407, the top value ‘ $\tilde{A}_{K_{max_top}}^i$ ’ = 87.5388, and the bottom value ‘ $\tilde{A}_{K_{max_bottom}}^i$ ’ = 33.7568, respectively.

Next, using Equations (11)–(13), a minimum fuzzy set ‘s12’ was obtained which is mentioned in Figure 8, where the fuzzy set-top value is ‘ $\tilde{A}_{K_{min_top}}^i$ ’ = −33.7568, the bottom value ‘ $\tilde{A}_{K_{min_bottom}}^i$ ’ = −87.5388 and the center absolute value ‘ $\tilde{A}_{K_{min_mid}}^i$ ’ = 55.2407, respectively.

The symmetric relationship between maximum and minimum fuzzy sets of $F_Sigmoid$ function is shown through Equations (11)–(13). The ' $\tilde{A}_{K_{min_mid}}^i$ ' is the center position of the fuzzy set ' $\tilde{A}_{K_{min}}'$ ', ' $\tilde{A}_{K_{min_top}}^i$ ' is the top position of the fuzzy set, and ' $\tilde{A}_{K_{min}}'$ ' and ' $\tilde{A}_{K_{min_b.tom}}^i$ ' is the bottom position for the fuzzy set of ' $\tilde{A}_{K_{min}}'$ ', respectively.

$$\tilde{A}_{K_{\min \text{ mid}}}^i = -\tilde{A}_{K_{\max \text{ mid}}}^i \quad (11)$$

$$\tilde{A}_{K_{\min \text{ } tom}}^i = -\tilde{A}_{K_{\max \text{ } h \text{ } tom}}^i \quad (12)$$

$$\tilde{A}_{K_{\min \text{ } b \text{ } tom}}^i = -\tilde{A}_{K_{\max \text{ } top}}^i \quad (13)$$

3. The Fuzzy Logic Processing

A fuzzy set assigns a membership grade, which selects an integer number from the interval of (0, 1). From the context of fuzzy sets framework emerges fuzzy logic which may address computational perception and cognition-related information: information that is unclear, imprecise, incomplete, or without sharp boundaries. In the development of intelligence, decision-making, identification, pattern recognition, optimization, and control systems, approaches based on fuzzy logic are available [38]. We have utilized fuzzy logic to select the best parameters during the controlling fuzzy sigmoid function activation interval. The fuzzy logic inference block diagram is shown in Figure 9, which consists of Fuzzification, Rulesets, Inference, and Defuzzification. The non-isometric triangular fuzzy sets are used to signify the relationship between input and output of fuzzy logic inference. Fuzzification, Inference, Rulesets, and Defuzzification are the four separate functions of the inference process, where the K_{max} and K_{min} are the inputs of fuzzification. The fuzzification mainly classifies the input values for corresponding fuzzy sets. The rule sets store all the regulations that are obtained by experiment. The inference combines the fuzzy sets and rules from Table 2 to determine the fuzzy α value. Then, defuzzification converts the fuzzy α value to the crisp α value (α_{crisp}) for the fuzzy sigmoid function.

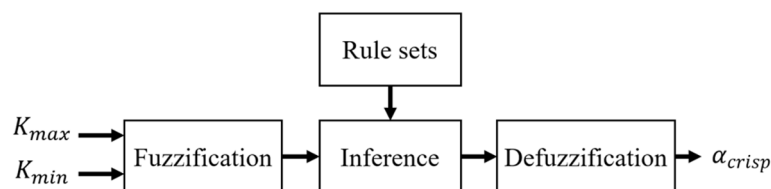


Figure 9. The fuzzy logic inference process.

Table 2. Rulesets. The diagonal symmetric relationship is highlighted in bold.

[illegible]

4. Fuzzification

The fuzzification mainly processes fuzzy information. Firstly, the fuzzy sets are defined for the inputs of K_{max} and K_{min} . The fuzzy information is calculated through membership grade [39], and the membership grade of x is denoted by $\mu(x)$. In addition, the membership grade for K_{max} and K_{min} in fuzzy logic which can be calculated by the functional Equations of (14) and (15). The membership grade of fuzzy sets for the K_{max} is denoted by the $\mu_{\tilde{A}_{K_{max}}^i}(K_{max})$ and K_{min} is denoted by $\mu_{\tilde{A}_{K_{min}}^i}(K_{min})$. \tilde{A} = Fuzzy set, and $\mu_{\tilde{A}}(x)$ = Membership grade.

$$\mu_{\tilde{A}_{K_{max}}^i}(K_{max}) = \begin{cases} \frac{\tilde{A}_{K_{max_top}}^i - K_{max}}{\tilde{A}_{K_{max_top}}^i - \tilde{A}_{K_{max_mid}}^i} & \tilde{A}_{K_{max_mid}}^i < K_{max} \leq \tilde{A}_{K_{max_top}}^i \\ 1 & \tilde{A}_{K_{max_mid}}^i = K_{max} \\ \frac{K_{max} - \tilde{A}_{K_{max_b.tom}}^i}{\tilde{A}_{K_{max_mid}}^i - \tilde{A}_{K_{max_b.tom}}^i} & \tilde{A}_{K_{max_b.tom}}^i \leq K_{max} < \tilde{A}_{K_{max_mid}}^i \end{cases} \quad (14)$$

$$\mu_{\tilde{A}_{K_{min}}^i}(K_{min}) = \begin{cases} \frac{K_{min} - \tilde{A}_{K_{min_top}}^i}{\tilde{A}_{K_{min_mid}}^i - \tilde{A}_{K_{min_top}}^i} & \tilde{A}_{K_{min_mid}}^i < K_{min} \leq \tilde{A}_{K_{min_top}}^i \\ 1 & \tilde{A}_{K_{min_mid}}^i = K_{min} \\ \frac{\tilde{A}_{K_{min_b.tom}}^i - K_{min}}{\tilde{A}_{K_{min_b.tom}}^i - \tilde{A}_{K_{min_mid}}^i} & \tilde{A}_{K_{min_b.tom}}^i \leq K_{min} < \tilde{A}_{K_{min_mid}}^i \end{cases} \quad (15)$$

If the maximum fully connected value of $K_{max} = 15.108$ and minimum fully connected value of $K_{min} = -12.153$, then Figure 10 depicts the membership grade calculation.

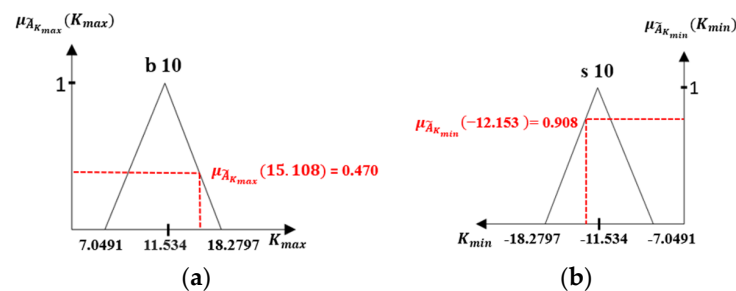


Figure 10. Fuzzification process (a) Maximum membership grade (b) Minimum membership grade.

5. Rulesets

The rules of fuzzy sets in Equation (16) have been customized by the experimental results then stored at inference. For the proceeding, “If” expresses antecedent and “Then” represents the consequent, whereas ‘ bi ’ and ‘ sj ’ are the sequences of maximum and the minimum fuzzy sets. The ‘ $\tilde{A}_{K_{max}}^{bi}$ ’ are the 13 maximum fuzzy sets represented by ‘ $b1$ to $b13$ ’ and ‘ $\tilde{A}_{K_{min}}^{sj}$ ’ are the 13 minimum fuzzy sets demonstrated by ‘ $s1$ to $s13$ ’, which represent a total of 169 rules that are shown in Table 2. The fuzzy set column ‘ $b13$ ’ and the row ‘ $s13$ ’ belongs to all ‘ $fzy1$ ’ fuzzy logic rules. On the contrary, the fuzzy set ‘ $b1$ ’ and ‘ $s1$ ’ only belong to ‘ $fzy13$ ’ logic. The rules table expresses a diagonal symmetric relationship between ‘ $\tilde{A}_{K_{max}}^{bi}$ ’ maximum fuzzy sets and ‘ $\tilde{A}_{K_{min}}^{sj}$ ’ minimum fuzzy sets.

Rule: If K_{max} belongs to $\tilde{A}_{K_{max}}^{bi}$ and K_{min} belongs to $\tilde{A}_{K_{min}}^{sj}$ then α_{crips} belong to $\tilde{A}_{\alpha}^{bi,sj}$.

$$i, j = 1, 2, \dots, 13 \quad (16)$$

6. Inference Process

The Inference combines the fuzzification outputs and rules to find the fuzzy α value. The Momani inference method [39] has been followed to obtain the antecedent and mem-

bership grade by the following Equations (17) and (18). The maximum and minimum membership grades are denoted by $(\mu_{A_{K_{max}}}^{bi}(K_{max}))$ and $\mu_{A_{K_{min}}}^{sj}(K_{min})$. The two inputs do 'min' value to obtain the result of the antecedent $(\tilde{C}^{bi, si})$, then $\tilde{C}^{bi, sj}$ do min with $\mu_{\alpha}^{bi, sj}(\alpha)$ to get the membership grade of the \tilde{A}_{α}^i on the i th rule. Finally, process the max value with membership grade of the $\tilde{A}_{\alpha}^{bi, sj}$ of all rules to obtain $\mu_{\tilde{A}_{\alpha}}(\alpha)$. Figure 11 illustrates the inference process and rules calculation.

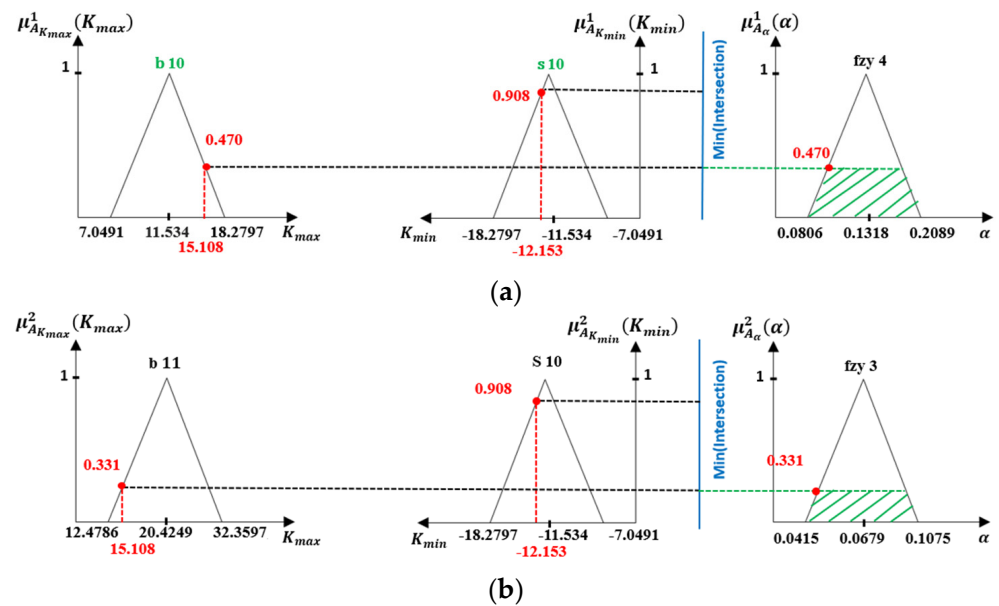


Figure 11. Mamani inference (a) Rules1 implementation (b) Rules 2 implementation.

Rule 1 : If K_{max} is b10 and K_{min} is s10 then α is fzy4

Rule 2 : If K_{max} is b 11 and K_{min} is s10 then α is fzy3

$$\tilde{C}^{b,s} = \min[\mu_{\tilde{A}_{K_{max}}}^{s}(K_{max}), \mu_{\tilde{A}_{K_{min}}}^{b}(K_{min})] \quad (17)$$

$$\mu_{\tilde{A}_{\alpha}}(\alpha) = \max[\min(\tilde{C}^{s,b}, \mu_{\tilde{A}_{\alpha}}^{s,b}(\alpha))] \quad (18)$$

7. Defuzzification

Defuzzification is the conversion procedure of a fuzzy set to transform a fuzzy output into a single crisp value. The defuzzification value in the fuzzy logic controller takes action to control the process. This approach offers a crisp value based on the center of gravity for the fuzzy set. There are several sub-areas inside of the total membership function employed for the combined control operation. The area and center of gravity or center of each sub-area are computed, and all these sub-areas are summed to identify the defuzzification value of a discrete fuzzy set. The inference comes with fuzzy sets that need to find the usable value through defuzzification. We have employed the method of the center of gravity for defuzzification by using Equation (19) where α_{crisp} is placed into α value for the fuzzy sigmoid function. α_{crisp} is the fuzzy sigmoid parameter, and $\mu_{A_{\alpha}}(\alpha)$ is the total membership grades of α_{crisp} . The defuzzification process and calculation shown in Figure 12.

$$\alpha_{crisp} = \sum \alpha \cdot \mu_{\tilde{A}_{\alpha}}(\alpha) / \sum \mu_{\tilde{A}_{\alpha}}(\alpha) \quad (19)$$

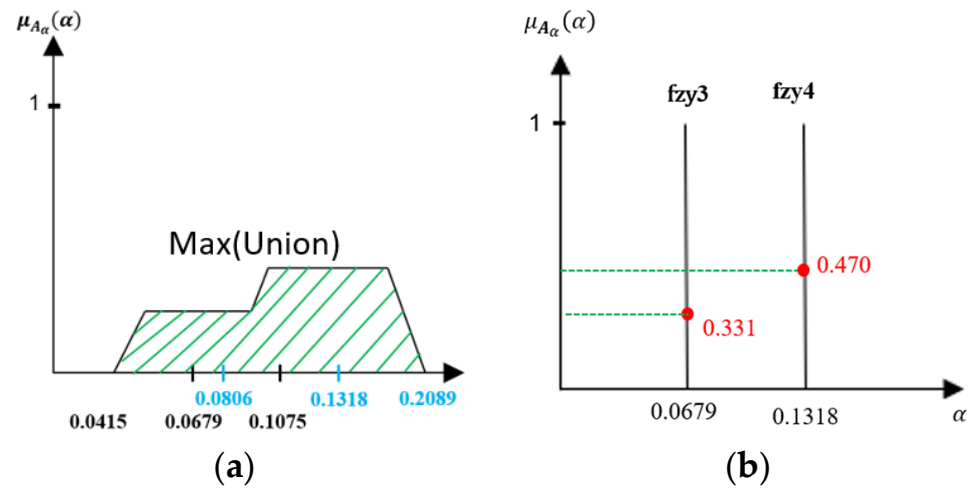


Figure 12. Defuzzification (a) Maximum Union of intersection and (b) Membership grades calculation.

3.3.3. Scale Choice

We proposed a Scale Choice (SC) layer that has training and testing labels, such as the training label shown in Figure 13. The $I = \{I_c, c = 1, \dots, C\}$ is the inputs of the FGSC block, $S = \{S_c, c = 1, \dots, C\}$ is the outputs of F_Sigmoid and $W = \{W_c, c = 1, \dots, C, 0 \leq W_c \leq 1\}$ is the weights of the scale choice. The S_c is responsible for recognizing the important channel feature of I_c . The main operation of S_c layers is the input feature I_c of each layer which is multiplied by the scale value, and then multiplied by the weight value W_c . The W_c learned the importance of each channel for the entire data set. If the $W_c = 1$, it means this channel feature is most important for the entire data set. When the $W_c = 0$, it means this channel feature is unimportant for the operation. These operations imposed important channel features through weights learning. The authors distinguished the importance of the operation choice mechanism that can skip the unimportant features on the test stage. The output of the scale choice layer is $Y = \{Y_{c^*}, c^* = 1, \dots, C^*\}$. The C^* is the remaining channel after the operation choice mechanism.

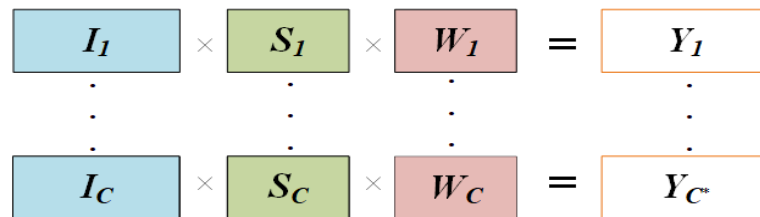


Figure 13. Training mode scale choice layer architecture.

- Operation Choice Mechanism

The scale choice recognizes the significant channel characteristics from the input channel to acquire the activation function scale values. If the W_c value is larger than the Jump-threshold (thr) value after training the Scale Choice layer, the feature values are multiplied by S_c and W_c . The ' thr ' level was set at 0.2 for this study. The following is the Algorithm 1 for the Scale Choice layer operation mechanism.

Otherwise, the small W_c stands for a lower feature value that has little effect on the following up to skip convolution.

Algorithm 1 Operation choice mechanism.

```

Input : SAC Block input  $I = \{I_c, c = 1, \dots, C\}$ ,
         $F\_Sigmoid$  output  $S = \{s_c, c = 1, \dots, C\}$ ,
Scale Choice layer weights  $W = \{w_c, c = 1, \dots, C, 0 \leq w_c \leq 1\}$ 
Output:  $b$ th SAC Block output  $Y_{c_b}^*$ 
1   Set  $thr = 0.2$ 
2   Set  $c_b^* = 0$ 
3   for  $c = 1: C$  do
4       if  $w_c > thr$  do
5            $Y_{c_b}^* = I_c \times s_c \times w_c$ 
6            $c_b^*++$ 
7       end
end

```

- Effects of Different Jump-threshold

Table 3 shows the test results of the various Jump-thresholds for PASCAL VOC datasets. The results are varying on the different jump thresholds. The result depicts that the FPS is low when the Jump-threshold is 0.08, and the accuracy is inadequate when the Jump-threshold is 0.3, so an appropriate threshold must look for better performance. Contrary to Jump-threshold 0.1 and 0.3, if the speed increases, the accuracy decreases, and vice versa. If the accuracy increases, the speed decreases. In this situation, the jump-threshold 0.2 has good speed and accuracy. Because we tend to increase the speed, the FGSC-SSD jump threshold is 0.2.

Table 3. The Result of the Different Jump-threshold. Better results are highlighted in Bold.

Model	Trainset	Jump-Threshold	mAP	FPS (PC)	FPS (Xavier)
FGSC-SSD	07 + 12	0.08	74.0	60.6	19.1
	07 + 12	0.1	74.1	63.4	19.6
	07 + 12	0.2	74.8	64.9	21.7
	07 + 12	0.3	73.2	65.2	20.4

- FGSC Block Weights distribution

We analyzed the weights of each FGSC block at the different scale choice layers for the PASCAL VOC datasets. Figures 14–18 represents the training weights distribution diagram. The weight values for FGSC1- Scale Choice rarely approached the 0, so the features of the FGSC1 block are most important. To have a clear understanding of the weight distribution of different layers, we set the jump threshold as 0.2 and make the figures on the percentage of skipping weight parameters in different scale choice layers. Table 4 shows the information of average weights and eliminating parameters for all FGSC blocks. The FGSC block 1 leaped 6.2% parameters, and the value of the average weight is 0.49. The FGSC block 5 skipped the 54% parameter, and the average weight value is 0.30. According to test results, the deep block leaps the higher percentage of the parameters and the lower percentage of the average weight value.

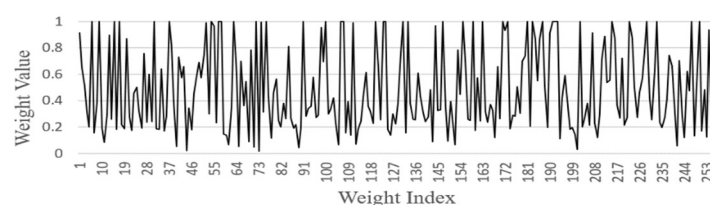


Figure 14. FGSC1-Scale choice layer weight.

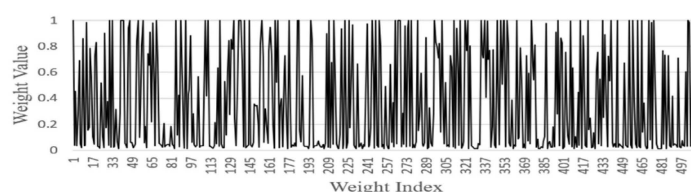


Figure 15. FGSC2-Scale choice layer weight.



Figure 16. FGSC3-Scale choice layer weight.

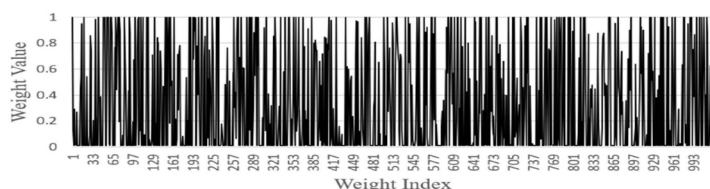


Figure 17. FGSC4-Scale choice layer weight.

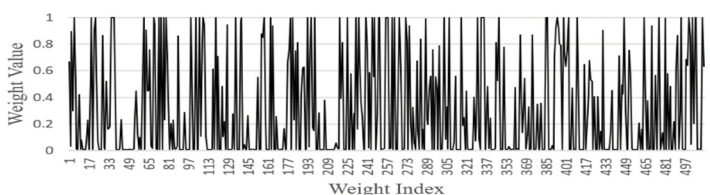


Figure 18. FGSC5-Scale choice layer weight.

Table 4. The Information of FGSC Block Parameters.

Block	The Ratio of Skipped Parameters
FGSC1	16/256 (6.2%)
FGSC2	258/512 (50.4%)
FGSC3	294/512 (57.4%)
FGSC4	541/1024 (53%)
FGSC5	276/512 (54%)

4. Vehicle Detection and Class Counting Process

4.1. Vehicle Detection Process

The operational function of vehicle detection and class counting overall process is illustrated in Figure 19. We set up passing detection lines and class counting windows separately for smooth and quick operation. The passing detection line detects whether the vehicle passes the line or not. If it passes the detection lines and reaches the center point of the class counting window, then the vehicle will be counted in a specific class. The passing detection lines were developed by a Gaussian Mixture Model (GMM) [40], which identifies whether the vehicle arrived and passed the detection line or not. We used background GMM color to build the passing detection line background pixels. When the car passed through the detection line, the pixel's color changed to the foreground GMM color. In addition, the background pixels set are 0 and the foreground pixels set are 1 so when the detection line GMM color pixels become 1 then the model considered the vehicle as passing the line. After passing the detection lines, when the vehicle reached the center of the class counting window then the algorithm counted the vehicles in particular classes.

In Figure 19, the forward passing detection line's color is still black because there is no vehicle, but the backward passing detection line's color has changed to the foreground pixels' color while a vehicle arrived at the detection line and vehicle class was counted.

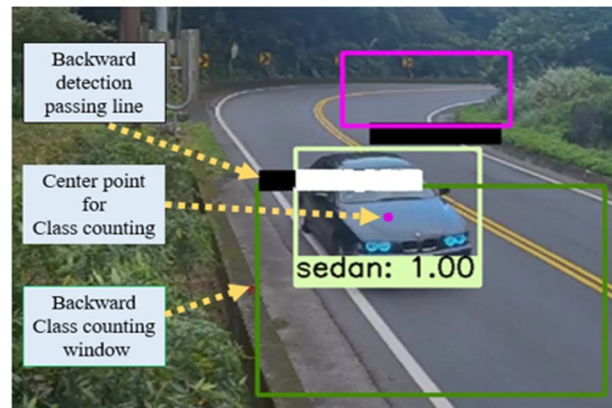


Figure 19. Vehicle detection and class counting process.

4.2. Detection Result at Different Weather Condition

To ensure safety issues and robust accurate object detection, the ability to recognize the object in different weather conditions is crucial for the deep learning algorithm. Under challenging weather, particularly rainy days, the performance of object detection algorithms might be reduced considerably. The FGSC-SSD deep neural network model can perform under various weather conditions. For example, in rainy weather, the visual conditions for vehicle detection and class counting experimental result are shown in Figure 20. The prediction accuracy for vehicle detection and class counting process is somewhat changed because of the effect of glare on visibility. Weather conditions include the meteorological changes of the environment due to precipitation including clear weather, rainwater, as well as cloudy weather. The performance of vision-based object detection methods drops significantly under rainy conditions because the camera sensors usually suffer from lowered gradient sizes, which cause the bounding boxes to vary their location and size throughout the detection process. In addition, the categorization values can be reduced, which indicates the increasing uncertainty that can, however, be neglected in near distances. In addition, it could be overcome by collecting a dataset with vehicles in rain scenes and training a state-of-the-art deep learning model using this dataset.



Figure 20. The detection result during the rainy day.

Extreme weather conditions have a significant impact on our everyday lives in a variety of ways. The object visibility is one of them, which is very limited in adverse weather conditions such as fog, ice, snow, and dust. In cold weather conditions, ice,

fog, dust, and light snow typically affect visibility. The grayscale image reflects color information to a large context and indirectly shows the feature information. We analyzed the influence of visibility in foggy weather on the accuracy of computer vision obstacles and detection. In this study, we considered visual conditions as the significant changes in the appearance of the experiment during the nighttime effect of brightness on perceptibility. However, the weather conditions are the barometric changes of the environment due to precipitation, including clear and cloudy weather at night. The experimental results during the nighttime with foggy weather show in Figure 21. We have trained the dataset with vehicles in foggy scenes and are training the deep learning model using this dataset.



Figure 21. The detection result at nighttime with foggy weather.

Machine learning vision technology has an impact on weather conditions. To perform a comparison at different weather conditions, the detection result shows that the nighttime with foggy weather has the lowest accuracy. Whereas the detection result during the rainy day achieved good performance. At the normal weather condition, our proposed FGSC-SSD deep neural network model reached the highest detection accuracy.

5. Experiment and Results Comparison

We utilized the PASCAL VOC datasets and the Benchmark BIT Vehicle dataset [41] to evaluate the performance of our proposed FGSC-SSD deep neural network model, trained on the “Caffe” [42] environment. The initial setup for the experiment: learning rate at 0.0001, the momentum as 0.9, and the batch size as 8. For weight initialization, we used a weight initialize strategy and stochastic gradient descent to optimize network weight parameters. To evaluate the performance of proposed FGSC-SSD deep network model in terms of detection, we considered mAP for accuracy and FPS for speed measurement.

Performance of the Test Datasets

Table 5 displays the performance of various deep neural network models for the PASCAL VOC (07 + 12) datasets. We have considered different GPU platforms to examine the efficiency of those models. On the Nvidia Titan Xp and Titan X platforms, the SSD model has 31.5 and 46 FPS which are better than other ME R-CNN, MSA-DNN, MFR-CNN, DF-SSD, and ACoupleNet approaches. Next, we replaced the platforms with 1080 Ti and Xavier, and the video base framework [43] SSD speed changed to 56 and 16.9, respectively. At the same stages, the YOLOv4 [44] has better precision of 78.9 and YOLOv3 s-highest FPS on 1080 Ti, but the proposed FGSC-SSD achieved the highest inference speed of 64.9 and 21.7 on those platforms, which is higher than the YOLOv3, YOLOv4, and SSD models. Compare to the SSD model, the proposed FGSC-SSD model has the highest accuracy and inference speed which approximates real-time processing speed for edge AI platforms.

Table 5. PASCAL VOC (07 + 12) Dataset Test Performance. The best FPS results of FGSC-SSD are highlighted in Bold.

Method	Platform	mAP	FPS
ME R-CNN	Titan Xp	72.2	13.3
MSA-DNN	Titan Xp	81.5	31.2
SSD	Titan Xp	81.7	31.5
MFR-CNN	Titan X	82.6	9.5
DF-SSD	Titan X	78.9	11.6
ACoupleNet	Titan X	83.1	6.9
SSD	Titan X	74.3	46
YOLOv3	1080 Ti	73.0	64.2
YOLOv4	1080 Ti	78.9	55.7
SSD	1080 Ti	73.7	56
FGSC-SSD	1080 Ti	73.8	64.9
YOLOv3	Xavier	73.0	19.3
YOLOv4	Xavier	78.9	13.6
SSD	Xavier	73.7	16.9
FGSC-SSD	Xavier	73.8	21.7

For the benchmark BIT-Vehicle datasets, Table 6 shows the test results of vehicle detection accuracy and inference speed for the YOLOv3, YOLOv4, SSD, and proposed FGSC-SSD deep neural network models on the platform of 1080 Ti and Xavier. Among those models, the YOLOv3 has the highest accuracy (96.3%), whereas the inference speed is second position on both platforms. However, the YOLOv4 and SSD models exhibit an accuracy 95.1 percent and 91.4 percent, respectively, and inference speed is 25.3 and 23.9 on the 1080 Ti platforms. In addition to Xavier platform, the YOLOv4 and SSD models achieved accuracy 95.1 percent and 91.4 percent same as the previous platform, but the FPS is 17.6 and 15.4, respectively. However, the proposed FGSC-SSD deep neural network model has reached highest inference proceeding speed of 26.8 and 21.7 on both platforms with high accuracy (95.5%), which is the second highest position for both platforms. In comparison to the baseline network SSD model, the proposed FGSC-SSD deep neural network model has achieved better accuracy and FPS on both platforms for the benchmark BIT-Vehicle datasets. Considering to the speed, the proposed FGSC-SSD deep neural network model reached the highest inference speed among those models.

Table 6. Benchmark BIT Vehicle dataset Test Performance. The best FPS results of FGSC-SSD are highlighted in Bold.

Method	Platform	mAP	FPS
YOLOv3	1080 Ti	96.3	26.2
YOLOv4	1080 Ti	95.1	25.3
SSD	1080 Ti	91.4	23.9
FGSC-SSD	1080 Ti	95.5	26.8
YOLOv3	Xavier	96.3	20.2
YOLOv4	Xavier	95.1	17.6
SSD	Xavier	91.4	15.4
FGSC-SSD	Xavier	95.5	21.7

Table 7 illustrated the accuracy performance for SSD and FGSC-SSD test results for specific vehicles for both models. Table 8 shows the result for video processing result on Nvidia Xavier. The FGSC-SSD model achieved an FPS of 38.4 which is faster than the backbone SSD model.

Table 7. Test Data Accuracy Results.

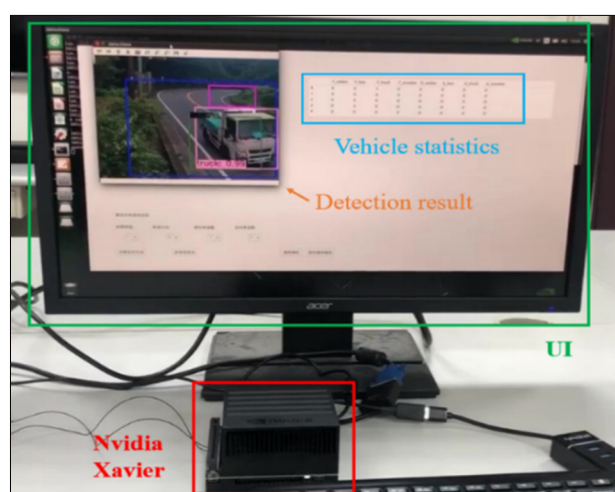
Method	Car	Bus	Truck	mAP
SSD	86.8	85.3	85.1	85.7%
FGSC-SSD	96.7	95.5	94.5	95.5%

Table 8. Practical Application Average FPS Results.

Method	Platform	FPS (Video 640 × 480)
SSD	Xavier	33.6
FGSC-SSD	Xavier	38.4

6. System Implementation

Figure 22 shows vehicle detection and class counting system implementation. We have utilized the edge AI platform for system implementation. In this experiment, the authors used actual mountain road videos for one day as test data. Table 9 depicts the result of vehicle detection and class counting. The detection and class counting accuracy for the car, bus, and truck are 96.7%, 95.5%, and 94.5%, respectively.

**Figure 22.** The Edge AI system implementation on Nvidia Xavier.**Table 9.** The Vehicle Class Counting Result from Actual video.

Vehicle Types	Vehicle Numbers	Correctly Counting	Error Counting	Correctly Counted
Car	61	59	2	96.7%
Bus	44	42	2	95.5%
Truck	36	34	2	94.5%

The actual results for vehicle detection and class counting from video are shown in Table 9. In total, 61 cars, 44 buses, and 36 trucks went through the vehicle detection lines, with 59 sedans, 42 buses, and 34 trucks properly counted, respectively, with 96.7 percent, 95.5 percent, and 94.5 percent accuracy. However, we noticed some errors in the class counting and reviewed the input data and results again. We found the causes of erroneous counting when two sedans of very old models passed the windows, but their images did not belong to the datasets. In addition, two recondition pickup vans went through detection lines that did not look like trucks. Furthermore, the two buses that crossed both detection lines that made an error for the class counting window.

7. Discussion

For considering the higher speed and real-time AIoT applications, we proposed the FGSC-SSD deep neural network model for performing an Edge AI platform. The proposed FGSC-SSD model can accomplish on edge computing for Real-Time IoT applications for vehicle detections. The fuzzy guided algorithm has reduced the errors and improved the vehicle detection rate and class counting. We have utilized a CPU Intel Core i7 8700k, GPU is Nvidia GTX 1080 Ti, CUDA version is 9.0, and cuDNN version is 7.0 to measure the performance by mAP and FPS. For the test experiment, we captured a real video from a mountain road in Taiwan. For future work, we intend to prepare more small-size vehicle images as training datasets to improve the performance capability and accuracy. Additionally, vehicle images datasets from different angles and various weather conditions can boost the performance of the proposed FGSC-SSD deep neural network model for vehicle detection and class counting system for all sorts of vehicles.

8. Conclusions

In this paper, the authors proposed an FGSC-SSD deep neural network model for vehicle detection and class counting with passing detection lines and class counting windows. The FGSC blocks comprise the global intensity, fully connected layers, fuzzy sigmoid function, and the scale choice layer. Hence, the fuzzy sigmoid function controls the activation interval and avoids unnecessary features falling into the saturation zone. The weight value is particularly significant for the scale choice layer, as it supports in learning the dataset's essential feature map and determining how to accomplish the counting optimization impact.

For performance, compared to the baseline network, the proposed FGSC-SSD has achieved higher proceeding speed and accuracy than the SSD model. For the authentic road video tests, vehicle detection and class counting accuracy of cars, busses, and trucks achieved 96.7%, 95.5%, and 94.5% accuracy, respectively. Moreover, the proceeding speed of the proposed FGSC-SSD model achieved 38.4 FPS, which is real-time proceeding speed on the Edge AI platform.

Author Contributions: Conceptualization, M.-H.S. and C.-J.L.; methodology, S.M.S.M.; software, S.M.S.M. and J.-X.Z.; validation, S.M.S.M. and J.-X.Z.; formal analysis, S.M.S.M. and J.-X.Z.; investigation, M.-H.S. and S.-C.H.; resources, M.-H.S.; data curation, C.-J.L.; writing—original draft preparation, S.M.S.M. and J.-X.Z.; writing—review and editing, S.M.S.M. and M.-H.S.; visualization, S.-C.H.; supervision, M.-H.S.; project administration, M.-H.S. and C.-Y.C.; funding acquisition, M.-H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Ministry of Science & Technology, Taiwan under the project No. 110-2221-E-224-052-MY2, and National Yunlin University of Science and Technology, Taiwan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available [37] PASCAL VOC (07 + 12) and Benchmark BIT Vehicle [41] open sources datasets were analyzed in this study. These data Sets can be found here: <https://paperswithcode.com/sota/active-object-detection-on-pascal-voc-07-12> (accessed on 2 August 2021) and BIT-Vehicle Dataset-Vehicle Model Identification Data Set-Programmer Sought.

Acknowledgments: This work has been supported by the Ministry of Science & Technology, Taiwan, and National Yunlin University of Science and Technology, Taiwan.

Conflicts of Interest: The authors have no conflict of interest to declare that are relevant to the content of this article.

References

- Ladislav, B.; Jaroslav, M. Changes in Road Traffic Caused by the Declaration of a State of Emergency in the Czech Republic—A Case Study. *Transp. Res. Procedia* **2021**, *53*, 321–328.
- Walid, B.; Hasan, T.; Hazem, H.R. Intelligent Vehicle Counting and Classification Sensor for Real-Time Traffic Surveillance. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 1789–1794.
- Cheng, H.Y.; Hsin, M.T. Vehicle Counting and Speed Estimation with RFID Backscatter Signal. In Proceedings of the IEEE Vehicular Networking Conference (VNC), Los Angeles, CA, USA, 4–6 December 2019; pp. 1–8.
- David, P.J.; David, S.K. An RFID-Enabled Road Pricing System for Transportation. *IEEE Syst. J.* **2008**, *2*, 248–257.
- Zhi, G.; Ruifang, Z.; Pengfei, W.; Xu, Y.; Hailong, Q.; Yazhe, T.; Bharath, R. Synergizing Appearance and Motion with Low-Rank Representation for Vehicle Counting and Traffic Flow Analysis. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 2675–2685.
- Honghong, Y.; Shiru, Q. Real-time Vehicle Detection and Counting in Complex Traffic Scenes Using Background Subtraction Model with Low-rank Decomposition. *IET Intell. Transp. Syst.* **2018**, *12*, 75–85.
- Lili, C.; Zhengdao, Z.; Li, P. Fast single-shot multi-box detector and its application on a vehicle counting system. *IET Intell. Transp. Syst.* **2018**, *12*, 1406–1413.
- Hou, X.; Zhang, L. Saliency Detection: A Spectral Residual Approach. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
- Yi, L.; Qiang, Z.; Dingwen, Z.; Jungong, H. Employing Deep Part-Object Relationships for Salient Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 1232–1241.
- Dingwen, Z.; Haibin, T.; Jungong, H. Few-Cost Salient Object Detection with Adversarial-Paced Learning. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 6–12 December 2020; pp. 1–12.
- Dingwen, Z.; Junwei, H.; Yu, Z.; Dong, X. Synthesizing Supervision for Learning Deep Saliency Network without Human Annotation. *IEEE Trans. Pattern. Anal. Mach. Intell.* **2020**, *42*, 1755–1769.
- Junwei, H.; Dingwen, Z.; Gong, C.; Nian, L.; Dong, X. Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey. *IEEE Signal Process. Magaz.* **2018**, *35*, 84–100.
- Wen, L.; Dragomir, A.; Dumitru, E.; Christian, S.; Scott, R.; Cheng, Y.F.; Alexander, C.B. SSD: Single Shot MultiBox Detector. In Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; 2016; pp. 21–37.
- Christian, S.; Wei, L.; Yangqing, J.; Pierre, S.; Scott, R.; Dragomir, A.; Dumitru, E.; Vincent, V.; Andrew, R. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
- Alex, K.; Ilya, S.; Geoffrey, E.H. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
- Mu, Y.; Brian, L.; Haoqiang, F.; Yuning, J. Randomized Spatial Pooling in Deep Convolutional Networks for Scene Recognition. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 27–30 September 2015; pp. 346–361.
- Ross, G.; Jeff, D.; Trevor, D.; Jitendra, M. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
- Ross, G. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
- Shaoqing, R.; Kaiming, H.; Ross, G.; Jian, S. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
- Kaiming, H.; Georgia, G.; Piotr, D.; Ross, G. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
- Hyungtae, L.; Sungmin, E.; Heesung, K. ME R-CNN: Multi-Expert R-CNN for Object Detection. *IEEE Trans. Image Process.* **2020**, *29*, 1030–1044.
- Hui, Z.; Kunfeng, W.; Yonglin, T.; Chao, G.; Fei, Y.W. MFR-CNN: Incorporating Multi-Scale Features and Global Information for Traffic Object Detection. *IEEE Trans. Vehicu. Techn.* **2018**, *67*, 8019–8030.
- Hai, W.; Yijie, Y.; Yingfeng, C.; Xiaobo, C.; Long, C.; Yicheng, L. Soft-Weighted-Average Ensemble Vehicle Detection Method Based on Single-Stage and Two-Stage Deep Learning Models. *IEEE Trans. Intell. Vehic.* **2021**, *6*, 100–109.
- Yousong, Z.; Chaoyang, Z.; Haiyun, G.; Jinqiao, W.; Xu, Z.; Hanqing, L. Attention CoupleNet: Fully Convolutional Attention Coupling Network for Object Detection. *IEEE Trans. Image Process.* **2019**, *28*, 113–126.
- Joseph, R.; Santosh, D.; Ross, G.; Ali, F. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
- Zhishuai, Z.; Siyuan, Q.; Cihang, X.; Wei, S.; Bo, W.; Alan, L.Y. Single-Shot Object Detection with Enriched Semantics. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5813–5821.
- Tsung, Y.L.; Priya, G.; Ross, G.; Kaiming, H.; Piotr, D. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.

28. Bichen, W.; Alvin, W.; Forrest, I.; Peter, H.J.; Kurt, K. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 446–454.
29. Hei, L.; Jia, D. CornerNet: Detecting Objects as Paired Keypoints. In Proceedings of the 15th European Conference, Munich, Germany, 8–14 September 2018; pp. 765–781.
30. Kaiyou, S.; Hua, Y.; Zhouping, Y. Multi-Scale Attention Deep Neural Network for Fast Accurate Object Detection. *IEEE Trans. Circuits Syst. Video Techn.* **2019**, *29*, 2972–2985.
31. Seelam, S.K.; Voruganti, P.; Nandikonda, N.; Ramesh, T.K. Vehicle Detection Using Image Processing. In Proceedings of the IEEE International Conference for Innovation in Technology (INOCON), Bengaluru, India, 6–8 November 2020; pp. 1–5.
32. Sheping, Z.; Dingrong, S.; Shuhuan, W.; Susu, D. DF-SSD: An Improved SSD Object Detection Algorithm Based on DenseNet and Feature Fusion. *IEEE Access* **2020**, *8*, 24344–24357.
33. Palo, J.; Caban, J.; Kiktova, M.; Cernicky, L. The Comparison of Automatic Traffic Counting and Manual Traffic Counting. In *Proceedings of the IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2019. [[CrossRef](#)]
34. Jie, H.; Li, S.; Gang, S. Squeeze-and-Excitation Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
35. Lewis, F.L.; Liu, K. Towards a Paradigm for Fuzzy Logic Control. In Proceedings of the Industrial Fuzzy Control and Intelligence (NAFIPS/IFIS/NASA '94), San Antonio, TX, USA, 18–21 December 1994; pp. 94–100.
36. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [[CrossRef](#)]
37. Mark, E.; Luc, V.G.; Christopher, K.I.W.; John, W.; Andrew, Z. The PASCAL Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338.
38. Skala, H.J. Fuzzy Concepts: Logic, Motivation, Application. In *Systems Theory in the Social Sciences*; Birkhäuser: Basel, Switzerland, 1976; pp. 292–306. Available online: https://doi.org/10.1007/978-3-0348-5495-5_13 (accessed on 21 July 2021). [[CrossRef](#)]
39. Timothy, J.R. *Fuzzy Logic with Engineering Application*, 3rd ed.; John Wiley & Sons Inc.: Hoboken, NJ, USA, 2010; pp. 90–94.
40. Chris, S.; Grimson, W.E.L. Adaptive Background Mixture Models for Real-Time Tracking. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, USA, 23–25 June 1999; pp. 246–256.
41. Jiani, X.; Zhihui, W.; Daoerji, F. A Solution for Vehicle Attributes Recognition and Cross-dataset Annotation. In Proceedings of the 13th International Congress on Image and Signal Processing, BioMedical Engineering, and Informatics (CISP-BMEI), Chengdu, China, 17–19 October 2020; pp. 219–224.
42. Yangqing, J.; Evan, S.; Jeff, D.; Sergey, K.; Jonathan, L.; Ross, G.; Sergio, G.; Trevor, D. Caffe: Convolutional Architecture for Fast Feature Embedding. In Proceedings of the 22nd ACM International Conference on Multimedia, New York, NY, USA, 3–7 November 2014; pp. 675–678.
43. Zhe, D.; Huansheng, S.; Xuan, W.; Yong, F.; Xu, Y.; Zhaoyang, Z.; Huaiyu, L. Video-based Vehicle Counting Framework. *IEEE Access* **2019**, *7*, 64460–64470.
44. Qi, C.M.; Hong, M.S.; Ling, Q.Z.; Rui, S.J. Finding Every Car: A Traffic Surveillance Multi-Scale Vehicle Object Detection Method. *Appl. Intell.* **2020**, *50*, 3125–3136.