Semantic Segmentation and Depth Estimation with RGB and DVS Sensor Fusion for Multi-view Driving Perception

Oskar Natan[1,2][0000−0003−3896−0448] and Jun Miura[1][0000−0003−0153−2570]

[1] Department of Computer Science and Engineering,
Toyohashi University of Technology, Aichi 441-8580, Japan
{oskar.natan.ao, jun.miura}@tut.jp
[2] Department of Computer Science and Electronics,
Universitas Gadjah Mada, Yogyakarta 55281, Indonesia
oskarnatan@ugm.ac.id

Abstract. In this research, we present a novel deep multi-task learning model to handle the perception stage of an autonomous driving system. The model leverages the fusion of RGB and dynamic vision sensor (DVS) images to perform semantic segmentation and depth estimation in four different perspectives of view simultaneously. As for the experiment, CARLA simulator is used to generate thousands of simulation data for training, validation, and testing processes. A dynamically changing environment with various weather conditions, daytime, maps, and non-player characters (NPC) is also considered to simulate a more realistic condition with expecting a better generalization of the model. An ablation study is conducted by modifying the network architecture to evaluate the influence of the sensor fusion technique. Based on the test result on 2 different datasets, the model that leverages feature maps sharing from RGB and DVS encoders is performing better. Furthermore, we show that our model can inference faster and have a comparable performance against another recent model. Official implementation code is shared at https://github.com/oskarnatan/RGBDVS-fusion.

Keywords: Sensor fusion · Semantic segmentation · Depth estimation · Multi-task learning · Driving perception.

1 Introduction

An autonomous driving system contains a lot of important elements, either on the hardware side or software side [10]. Specifically, on the software side, there are four main stages that must be considered to achieve a fully autonomous driving operation, there are environment perception, localization, planning, and control [14]. As the first stage, perception has always been a challenging task in developing the foundation of the complex autonomous driving system. The system needs to understand what kinds of objects are showing up on cameras and their relative distance from the ego vehicle. Once the necessary information

This is the accepted version of the manuscript.
The final authenticated version is available online at https://doi.org/10.1007/978-3-031-02375-0_26.
Pattern Recognition. ACPR 2021. Springer Lecture Notes in Computer Science

is ready, another task such as localization, path planning, and actuator control can be done. An autonomous driving vehicle is usually equipped with several sensors, mainly RGB cameras to capture multiple views of surroundings [11]. When it comes to computer vision problems, deep learning algorithm, especially convolutional neural network (CNN) has been proofed as the state-of-the-art by plenty of research [23]. For instance, a CNN-based hierarchical multi-scale attention model [26] improved by Borse et al. [1] is currently holding the highest intersection over union (IoU) score of 85.6% in performing semantic segmentation on a well-known autonomous driving dataset Cityscapes [5]. Another example comes from the depth estimation task which is related to the estimation of relative distance on each pixel of an image. Ranftl et al. [20] currently hold the best depth estimation performance on NYU-Depth V2 dataset [17] with root mean squared error of 0.357. Therefore, it would be the right decision if CNN is used as the main algorithm to deal with various perception tasks.

Currently, there are plenty of important tasks in the field of autonomous driving perception such as semantic segmentation and depth estimation as mentioned earlier. Developing a single-task deep learning model to handle each task can be costly and inefficient [30]. Instead of split the task, we develop multi-task learning (MTL) model that follows an encoder-decoder fashion with additional skip connections to deal with multiple perception tasks simultaneously. Then, another challenge comes from multi-input and multi-modal processing where multiple data need to be processed together to achieve a better understanding. To address this issue, a sensor fusion technique can be adopted where various sensors are deployed to provide more various data in representing the environmental condition [7]. In this research, we placed 8 sensors composed of RGB and DVS cameras in 4 different positions on the ego vehicle to capture front (F), left (L), right (R), and back (B) views. DVS sensors are used to support RGB cameras in providing more information as it is susceptible to illumination variations, especially during night time [16]. With various data modalities taken as inputs, the model is expected to perform better in a dynamically changing environment. In this research, the model is developed entirely from scratch without transfer learning from any pre-trained models so that the architecture is highly customizable. Moreover, we conduct the experiment on 2 different datasets gathered from CARLA simulator [6] to strengthen our findings. Based on the aforementioned approaches, the novelty of this research can be listed in the following points.

- We present a deep MTL model that has input-specified encoders and task-specified decoders. The model processes and fuses multiple inputs of RGB and DVS images to perform multiple views of semantic segmentation and depth estimation in one forward pass. Since the model is developed entirely from scratch, the architecture is highly extendable to retrieve more inputs.
- We study the influence of sensor fusion between RGB and DVS cameras. Based on the ablation experiment, our model gains more improvement when both RGB and DVS images are fused. Moreover, we also perform a comparative study with another recent MTL model to clarify their performance based on the metric score and inference speed.

2 Related Work

To date, a lot of research in driving perception has been done. In this section, we explain several kinds of related research which are also inspiring our works.

2.1 Deep Multi-task Learning

In deep learning, the process of learning to perform several tasks simultaneously is called multi-task learning. MTL aims to leverage shared feature maps during the training process to boost the performance of each task [21]. There is plenty of studies in the MTL area that is applied to the autonomous driving vehicle problems such as conducted by Teichmann et al. [27] where a deep learning model called MultiNet is used to perform several perception tasks such as road segmentation, vehicle detection, and street classification simultaneously. The model works well, however, it needs improvement in recognizing more crucial and various objects on the road. Koci et al. [12] presented a network architecture called J-Net that processes RGB images to perform various control tasks such as controlling the steering wheel, speed, brake, etc. However, the simulation condition still needs to be improved to test the model generalization. These issues are solved by Cipolla et al. [4] where they develop a semantic segmentation model to recognize more various kinds of objects in Cityscapes dataset [5]. It also performs instance segmentation and depth estimation by creating a branch of task-specified decoder for each task. However, it would be better if the model can take multiple inputs with multiple data modalities so that it can be applied to multi-view systems for a better understanding.

2.2 Sensor Fusion in Deep Learning

A study of multiple inputs processing on an autonomous driving perception model has been done by Hane et al. [9] where several cameras are placed in several different positions on the ego vehicle. Therefore, the model will have a better capability in understanding the environmental condition. Although it has more views of RGB images, the model may still fail during nighttime or heavy rain due to poor illumination conditions. Thus, it needs to be combined with another kind of sensor such as a DVS camera as an alternative in providing surrounding information, especially in performing an active perception [16] [29]. Hence, the idea of using the DVS camera can also be adopted in solving an autonomous driving perception problem. To handle various data modalities, Nobis et al. [18] have shown that a deep learning model can be used to process multiple sensor data by fusing extracted feature maps from each input modality. Nobis et al. develop an object detection model called CameraRadarFusionNet (CRF-Net) that processes camera and radar data to get a better result on nuScene dataset [2]. CRF-Net provides a specific encoder for each input data and fuses extracted feature maps into the Feature Pyramid Network [15] to perform bounding boxes regression and classification in one forward pass.

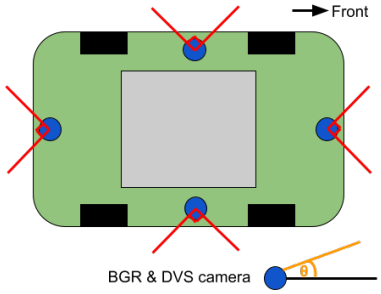

Fig. 1: Sensor placement on the ego vehicle. Four pairs of RGB and DVS cameras are mounted in 4 different positions. The camera's vertical angle of view θ is set to 20^o to catch a better view of the surroundings. The red lines on each camera represent the horizontal field of view which is set to 90^o.

In this research, we consider adopting several ideas shown in the aforementioned research. First, we utilize DVS cameras to provide more information as it is more robust against poor illumination problem [16] [29]. However, instead of being used to replace RGB cameras, DVS cameras are utilized as support sensors. Therefore, we also adopt the idea of sensor fusion shown by Nobis et al. [18], especially on its strategy in providing an input-specified encoder to process each input modality and fuse them several times at several points in the architecture. Finally, we adopt the task-specified decoder inspired by Cipolla et al. [4] to create two branches of decoders that perform semantic segmentation and depth estimation simultaneously. Hence, a complete deep learning model that processes and fuses multiple inputs to perform multiple tasks can be achieved.

3 Methodology

In this section, we describe the generation of simulation data using CARLA simulator with a lot of settings on weather conditions, NPCs, sensors, and many more. Finally, the network architecture is explained along with hyperparameter tuning and experiment setup including the loss and metric formulation.

3.1 Dataset and Input/Output Representation

In this research, CARLA simulator [6] is used to generate thousands of simulation data for training, validation, and testing sets retrieved from F, L, R, B views of the ego vehicle as shown in Figure 1. Various kinds of weather conditions (sunny, rainy, foggy, cloudy) and day times (morning, noon, evening, night) are considered to vary the simulation conditions. Besides that, we also use 2 different CARLA maps to ensure the fairness of model validation and testing. Then, numerous pedestrians and other vehicles such as cars, trucks, bikes, and motorcycles are also spawned to simulate more realistic conditions. Therefore, the model is expected to have a better generalization.

As for the model inputs/outputs (I/O), both RGB and DVS images along with their semantic segmentation and depth estimation ground truths. To evaluate the model performance with considering a fair experimental study, we create 2 different sets of data named set A and B which are taken from 2 different CARLA maps named town01 and town02. Based on the CARLA documentation, town01 and town02 are different but have similar characteristics. Therefore, we can make a fair experiment by separating the training set from validation and testing sets completely by differentiating the map. In set A, 4000 pairs of I/O from town01 are used for training, 1000 pairs of I/O from town02 for validation and another 1000 pairs of I/O from different regional areas in town02 are used for testing. Meanwhile, in set B, we retrieve the same amount and ratio of data but with a different map configuration where town02 is used for training, and town01 is for validation and testing. With this scenario, a fair experiment can be achieved where the road situation for each set are totally different.

During the data generation process, we retrieve RGB images as $I_{RGB} \in \mathbb{R}^{128\times128\times3}$ which represent the height, width, and RGB channels with 8-bit value on each pixel. Meanwhile, the original form of DVS images are arrays $A_{DVS} \in \mathbb{R}^{N\times4}$ which represents the total N number of 4-elements 1D array retrieved in one simulation step. The element for each array is composed of timestamp, x-position, y-position, and polarization. The timestamp is used for data synchronization with RGB images and ground truths. Then, the x and y-position show the pixel location considered to have a brightness change. The x,y-position has a maximum range of 128. Meanwhile, the polarization can be positive or negative based on the brightness change. Depends on how many pixels are changing, the total number of N can be different for each DVS array A_{DVS}. Thus, in order to fix the number of array elements in A_{DVS} and also to match with the input layer of the network architecture, we pre-process DVS arrays to be DVS images $I_{DVS} \in \mathbb{R}^{128\times128\times2}$ which represent the height, width, and polarization channels. The x,y-position takes place on the height and width of I_{DVS}, while the polarization takes place on the channel where the first channel is used for positive polarization and the second channel is used for negative polarization. We set the pixel that has polarization to have a full 8-bit value and otherwise 0. Mathematically, it can be written as in (1).

$$I_{DVS_i} = \begin{cases} 255 & \text{if } I_{DVS_i} \text{ is polarized} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where I_{DVS_i} is the positive or negative polarization of i_{th} pixel of the DVS image I_{DVS}. As for ground truths, semantic segmentation images are retrieved from the CARLA simulator following the RGB color pallets in the cityscapes dataset [5]. We use 6 classes of objects (poles, road, road lanes, sidewalks, pedestrian, vehicles) that are considered as important objects in the area of driving perception. We handled the semantic segmentation mask for each class by separating each unique colormap in $I_{segmentation} \in \mathbb{R}^{128\times128\times3}$ and store them into a tensor $I_{mask} \in \mathbb{R}^{128\times128\times6}$ where each element has the value of 0 or 1 depends on what kind of semantic class appear on each pixel in $I_{segmentation}$.

Meanwhile, the ground truth of the depth estimation task can be obtained by simply read the logarithmic depth images as $I_{depth} \in \mathbb{R}^{128\times128\times1}$ which represent the height, width, and 1 channel of 8-bit logarithmic depth value. For the training purpose, each element in all input and output tensors are normalized between 0 to 1 so that each element will have the same influence. Finally, the channel for each input and output tensors are moved to the first axis as needed by PyTorch deep learning framework [19]. Therefore, RGB and DVS inputs size become $(3 \times 128 \times 128)$ and $(2 \times 128 \times 128)$ respectively. Meanwhile, semantic segmentation and depth estimation outputs size become $(6 \times 128 \times 128)$ and $(1 \times 128 \times 128)$ respectively.

3.2 Network Architecture

As mentioned in the previous subsection, we use PyTorch deep learning framework [19] to build the model along with other python packages. The visualization of the model architecture can be seen in Figure 2. The model is following the encoder-decoder style [28] with additional skip connections inspired from UNet paper [22]. Each feature map in both RGB and DVS encoders is connected to its symmetric feature maps in both depth estimation and semantic segmentation encoders. With this configuration, each encoder can act as a supporter for one another. For example, when the illumination is very poor (e.g. night) and RGB cameras are failed to capture enough information of the surroundings, the network can learn how to leverage extracted information mainly from the DVS encoder. On the other hand, when the car is not moving (e.g. at the crossroads) and there is no enough information as the brightness change is very rare, the network can learn how to rely more on the RGB encoder. In this architecture, each convolutional block on encoder and decoder has 2 times of $((3\times3)$ convolutional layer + batch normalization + ReLU activation). Meanwhile, each convolutional block on the bottlenecks has 3 times of them to extract more information from all views concatenation. Then, they are followed by (2×2) max-pooling on the encoder side and (2×2) bilinear upsampling on the decoder side. To deal with the overfitting issue, several dropout layers with $p = 0.5$ are placed on the bottleneck [24]. Finally, a pointwise (1×1) convolutional layer is used to reduce the channel number of feature maps to match with the ground truth size. Then, it is followed by a sigmoid activation for semantic segmentation and a ReLU activation for depth estimation.

In order to discover the advantage of sensor fusion on an MTL model, an ablation study is performed during the experiment. First, we remove the DVS input block (blue) so that the model only processes RGB images to perform semantic segmentation and depth estimation. We refer to this model as A0 where only RGB images are fed into the network. Then, on the second model named A1, DVS inputs are added so that the model is processing 4 pairs of RGB and DVS images. Furthermore, feature maps from DVS encoders are concatenated to the semantic segmentation and depth estimation decoders as well as feature maps from RGB encoders.

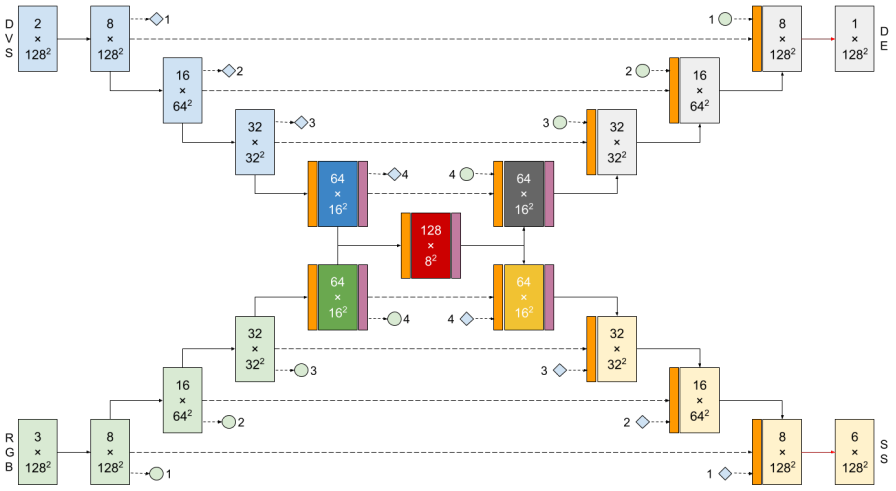

Fig. 2: Network architecture. Blue and green boxes represent the inputs and feature maps for each view (F, L, R, B). Dark blue and dark green boxes are the concatenation across all views, while the dark red box is the concatenation of all feature maps. Then, grey and yellow boxes represent feature maps and outputs for each view of depth estimation (DE) and semantic segmentation (SS). 5 boxes in the center are considered as bottlenecks where a dropout layer (purple) is applied for each. Dark grey and dark yellow boxes are the specific bottlenecks for each task. Solid lines represent the convolution block, while dashed lines represent the skip connection and concatenation (orange boxes). Each dashed line is connecting the feature map on encoders with its symmetric feature map on decoders. Denoted with numbered small blue squares and green circles, both encoders are used to support both tasks decoders. Finally, red lines represent the final pointwise convolution followed with an activation function.

3.3 Experiment Setup

We set the batch size to 16 so that there will be 250 steps of weights update in one epoch. Kaiming initialization method [8] is used to initialize entire model weights so that the model can be converged faster. Then, a standard Stochastic Gradient Descent (SGD) [25] with momentum $\beta = 0.9$ is used to train the model. For semantic segmentation loss, pixel-wise binary cross-entropy (BCE) (2) combined with dice loss (3) are calculated together to allow loss diversity.

$$\mathcal{L}_{BCE} = \frac{1}{4}\sum_{t=1}^{4} -\frac{1}{N}\sum_{i=1}^{N} y_{ti} \times log(\hat{y}_{ti}) + (1 - y_{ti}) \times log(1 - \hat{y}_{ti}) \tag{2}$$

$$\mathcal{L}_{dice} = \frac{1}{4}\sum_{t=1}^{4} 1 - \frac{2|\hat{y}_t \cap y_t|}{|\hat{y}_t| + |y_t|} \tag{3}$$

We average all losses from all views (F, L, R, B). N is total elements in the predicted output tensor of task t (denoted with $\hat{y}_t$) which is the same as the ground truth y_t with the size of $(6 \times 128 \times 128)$. Meanwhile, y_{ti} is ground-truth value of i^{th} pixel in y_t and $\hat{y}_{ti}$ is predicted value of i^{th} pixel in $\hat{y}_t$ after sigmoid activation. Then, as for the depth estimation loss, Huber loss (4) is used since it has some benefits compared to mean absolute error (MAE) that constantly have a large gradient and mean squared error (MSE) which is not robust against outliers. Huber loss is suitable for any regression-based task as it curves around the minima and is robust against outliers.

$$\mathcal{L}_{huber} = \frac{1}{4}\sum_{t=1}^{4}\frac{1}{N}\sum_{i=1}^{N} z_{ti}, \tag{4}$$

where z_{ti} is given by (5).

$$z_{ij} = \begin{cases} 0.5(\hat{y}_{ij} - y_{ij})^2 & \text{if } |\hat{y}_{ij} - y_{ij}| < \alpha \\ \alpha(|\hat{y}_{ij} - y_{ij}| - 0.5\alpha) & \text{otherwise} \end{cases} \tag{5}$$

Similar to the semantic segmentation, y_{ti} is the ground truth value of i^{th} pixel in y_t and $\hat{y}_{ti}$ is the predicted value of i^{th} pixel in $\hat{y}_t$ after ReLU activation. However, the size of $\hat{y}_t$ and y_t is only $(1 \times 128 \times 128)$ since there is only 1 channel containing the normalized logarithmic depth value. Then, we set $\alpha = 0.5$ as the threshold for Huber loss to start to curve around the minima. Finally, the total loss can be calculated as in (6).

$$\mathcal{L}_{total} = (\mathcal{L}_{BCE} + \mathcal{L}_{dice}) + 1.5 \times \mathcal{L}_{huber} \tag{6}$$

Huber loss is weighted with a constant of 1.5 to balance the huge value computed by the semantic segmentation loss function which is composed of BCE and dice loss. Therefore, the model is expected to not losing its learning focus on the depth estimation task. During the training process, weight decay w_d is also used to penalize the model complexity and to prevent overfitting [13]. Thus, the final total loss can be calculated as in (7).

$$\mathcal{L}_{total} = \mathcal{L}_{total} + w_d \times \Sigma w^2, \tag{7}$$

The sum-squared of all model weights (Σw^2) can be very large. Thus, we set a small value of $w_d = 0.0001$ so that it will not affect the total loss too much, as we want the model to learn more from depth estimation and semantic segmentation losses. In order to evaluate model performance, we also compute metric functions composed by average IoU (8) and average MAE (9).

$$IoU = \frac{1}{4}\sum_{i=1}^{4}\frac{|\hat{y}_t \cap y_t|}{|\hat{y}_t| \cup |y_t|} \tag{8}$$

$$MAE = \frac{1}{4}\sum_{i=1}^{4}\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_{ti} - y_{ti}| \tag{9}$$

As shown in (8) and (9), the IoU and MAE are averaged over all views which means that we evaluate the model performance globally. Finally, we also average the loss and metric on all batches for both training and validation sets on each epoch to monitor and evaluate the model performance. In accordance with the fast and smooth update of the model weights, we set the initial learning rate $\eta_0 = 0.1$ and reduce it by half gradually if the validation total metric (TM) (10) is not decreasing in 5 epochs in a row. The learning rate reduction will stop if it hits the minimum learning rate of $\eta_{min} = 0.00001$.

$$TM = MAE + (1 - IoU) \tag{10}$$

To prevent unnecessary computational cost, an early stopping strategy is used to stop the training process if there is no drop in the validation total metric in 35 epochs in a row. Therefore, the total epochs might be different for each model.

4 Result and Discussion

In this section, an ablation study is performed to study the effect of adding DVS images as the model inputs. In this study, we create 2 different model variants where the first one only takes RGB images (A0) while the other one takes both RGB and DVS images (A1) as described in Subsection 3.2. Then, to clarify the model performance, we also conduct a comparative study against another MTL model named W-Net [3] that performs the same tasks. Since there are 4 views around the ego vehicle, we create a W-Net model for each view. The evaluation is conducted by comparing metric scores on the validation and testing set as described in Subsection 3.3. Moreover, we also compare the inference speed to strengthen our justification. Finally, as the qualitative study, we provide several samples of inference results on both test sets.

4.1 Performance Gain by Sensor Fusion

RGB images are usually used as the only input when dealing with semantic segmentation and depth estimation tasks. However, in this research, we study the influence of providing DVS images as the input and fused together with RGB images in the network architecture to leverage the extracted information. As mentioned in Subsection 3.2, in order to evaluate the contribution of DVS images, we first remove the DVS input block on the Figure 2 and named the model as A0 model, then compare its performance with the A1 model that has both RGB and DVS input blocks on its architecture. To perform fair testing and comparison, we use 2 different datasets as described in Subsection 3.1. The comparison of validation total metric, semantic segmentation IoU, and depth estimation MAE on set A and set B can be shown in Figure 3. The A1 model has a lower total metric score compared to the A0 model on both validation sets. During the validation process, the A1 model has the record of the lowest validation TM of 0.148 (set A) and 0.190 (set B).

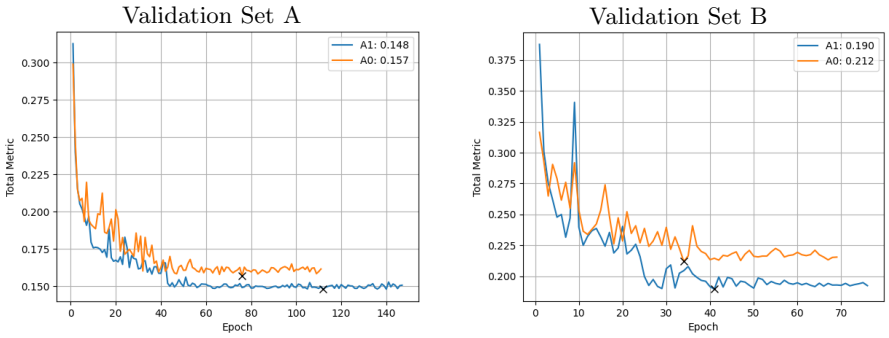

Fig. 3: Performance comparison during validation process

Note: Black × mark means the best score among all epochs.

Table 1: Model Performance Comparison on Test Sets

Dataset	Model	TM Score	Depth MAE	Segm. IoU	FPS
Test A	A0	0.196 ± < 0.001	**0.056 ± <0.001**	0.860 ± <0.001	**100**
	A1	0.188 ± < 0.001	0.059 ± <0.001	0.871 ± <0.001	83
	W-Net [3]	**0.166 ± < 0.001**	0.057 ± <0.001	**0.891 ± <0.001**	58
Test B	A0	0.193 ± < 0.001	0.038 ± <0.001	0.845 ± <0.001	**104**
	A1	**0.186 ± < 0.001**	**0.035 ± <0.001**	**0.849 ± <0.001**	84
	W-Net [3]	0.220 ± 0.001	0.038 ± <0.001	0.818 ± <0.001	57

The uncertainty on each prediction score is measured by calculating the variance over 1000 inference results. Meanwhile, the speed test is conducted on the same NVIDIA RTX 3090 with batch size = 1 and calculated in frame per second (FPS). The FPS difference on both datasets is caused by the fluctuating GPU condition.

Based on the inference result on testing sets shown in Table 1, the A1 model also has lower TM scores of 0.188 (set A) and 0.186 (set B). Considering that the A1 model has a better score in all validation and testing sets, it can be said that DVS is giving a positive influence on the model performance. However, as a result of having more encoders to process DVS data, the A1 model inference slower than the A0 model with an FPS rate of around 83 to 84. Meanwhile, the qualitative result can be seen in Figure 4 where both A0 and A1 models are deployed in the night (test set B: town01, left) and cloudy day (test set A: town02, right). The A1 model seems to have a better result compared to the A0 model. The A1 model is more stable in segmenting rare objects such as poles and the small appearance of surrounding vehicles, especially during a poor illumination condition (night) as it can leverage the information provided by the DVS. However, both model seems comparable in the depth estimation task.

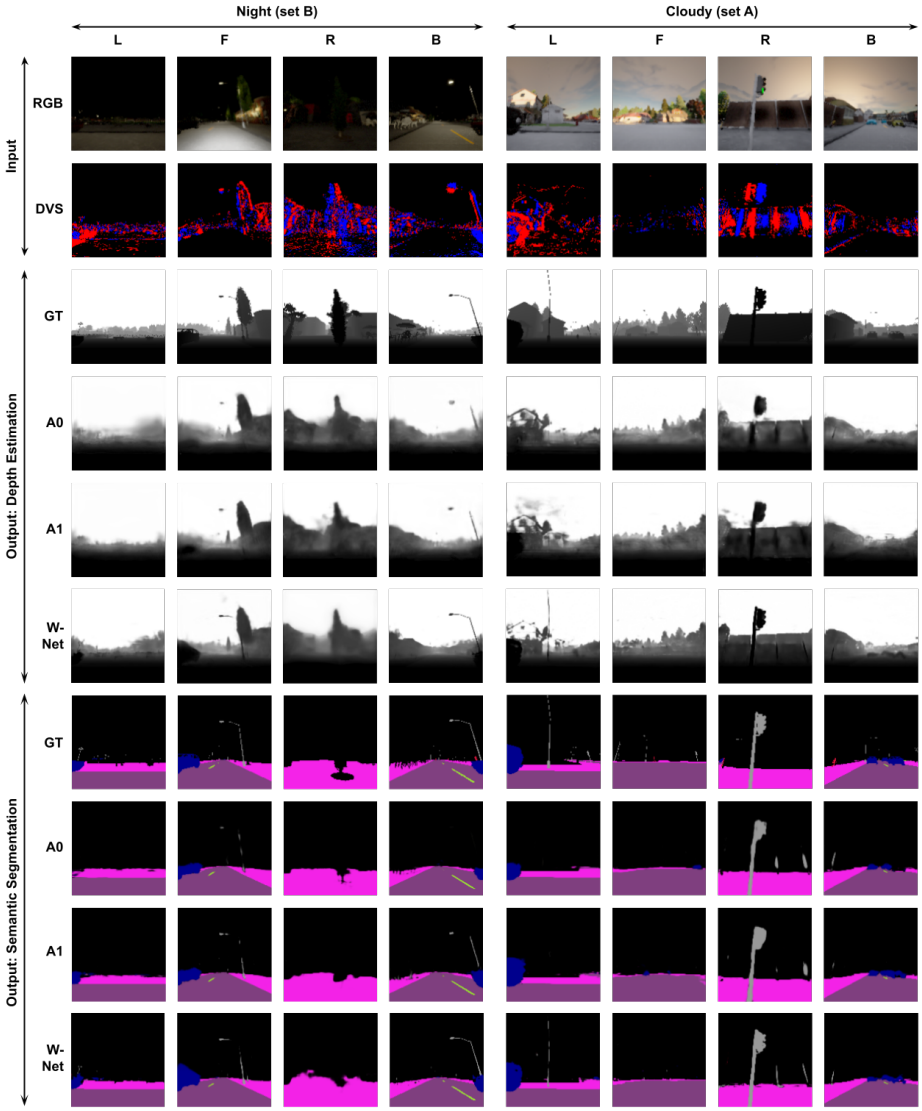

Fig. 4: Inference on test images

Note: F (front); L (left); R (right); B (back); GT (ground truth).

4.2 Comparison with Another Model

As mentioned in Section 1, a further comparative study against another MTL model is conducted to clarify the model performance. In this research, we compare our model with W-Net [3] which is composed of 2 serially connected UNet models [22]. W-Net uses its first UNet block to perform semantic segmentation first. Then, the prediction is concatenated with the RGB image as the input for the second UNet block to perform depth estimation. Therefore, W-Net is able to perform both semantic segmentation and depth estimation simultaneously in one forward pass. For a fair comparison, we follow the training configuration described in the W-Net paper to train the model using our datasets. Following the mathematical formulas described in Subsection 3.3, the comparison of metric scores can be seen in Table 1. To be noted, the metric score is averaged across all views as there are 4 independent W-Net models in total.

Based on Table 1, both A1 and W-Net models can be said to be comparable to each other. On test set A, W-Net performs better than the A1 model with a lower TM score of 0.166. Meanwhile, on test set B, the A1 model is surpassing W-Net with a lower TM score of 0.186. Then, as shown in Figure 4, W-Net seems to have a better result, especially when the illumination is enough (set A) as it has much more layers compared to the A1 model. W-Net can estimate and segment very thin objects such as light poles on both left images of depth estimation and semantic segmentation. However, in the term of inference speed, A1 model is still better with an FPS rate of more than 80 on both datasets. Meanwhile, W-Net only achieves an FPS rate of below 60 when tested with the same device. Therefore, even though the TM score is comparable, it can be said that a single A1 model is more preferable as it can perform faster inference compared to the combination of 4 W-Net models.

5 Conclusion and Future Work

In this research, we discover the usefulness of the sensor fusion of RGB and DVS cameras by comparing 2 model variants: A0 (without DVS) and A1 (with DVS). Then, a comparative study against another model named W-Net is conducted to clarify the model performance.

Based on the ablation study, we conclude that fusing both RGB and DVS images will boost the overall model performance since the model can take more distinctive information from both RGB and DVS encoders. From the test result on both datasets, the total metric scores are constantly lowered from 0.196 to 0.188 (set A) and 0.193 to 0.186 (set B). Moreover, the A1 model can deal with a poor illumination issue compared to the A0 model that fails to segment surrounding vehicles correctly. Considering that the A1 model is gaining more improvement, it can be said that taking both RGB and DVS images is better to get a more clear scene understanding. Furthermore, the A1 model still maintains to has comparable performance compared with the combination of 4 W-Net models. Considering the further model deployment, the A1 model is more preferable as it can inference faster with an FPS rate above 80.

In the future, the research can be extended to study the network architecture modification, especially on its convolutional blocks. There are plenty of well-known convolutional blocks that are able to improve the model performance further. Then, a study on loss function formulation is also an interesting study, especially in dealing with rare objects issue.

References

1. Borse, S., Wang, Y., Zhang, Y., Porikli, F.: InverseForm: A loss function for structured boundary-aware segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5901–5911 (2021)
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11618–11628 (2020)
3. Cantrell, K., Miller, C., Morato, C.: Practical depth estimation with image segmentation and serial U-Nets. In: Proceedings of the International Conference on Vehicle Technology and Intelligent Transport Systems. pp. 406–414 (2020)
4. Cipolla, R., Gal, Y., Kendall, A.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7482–7491 (2018)
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3213–3223 (2016)
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the Annual Conference on Robot Learning. pp. 1–16 (2017)
7. Fayyad, J., Jaradat, M.A., Gruyer, D., Najjaran, H.: Deep learning sensor fusion for autonomous vehicle perception and localization: A review. Sensors **20**(15) (2020)
8. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1026–1034 (2015)
9. Hne, C., Heng, L., Lee, G.H., Fraundorfer, F., Furgale, P., Sattler, T., Pollefeys, M.: 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. Image and Vision Computing **68**, 14–27 (2017)
10. Kato, S., Takeuchi, E., Ishiguro, Y., Ninomiya, Y., Takeda, K., Hamada, T.: An open approach to autonomous vehicles. IEEE Micro **35**(6), 60–68 (2015)
11. Khatab, E., Onsy, A., Varley, M., Abouelfarag, A.: Vulnerable objects detection for autonomous driving: A review. Integration **78**, 36–48 (2021)
12. Kocic, J., Jovicic, N., Drndarevic, V.: An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. Sensors **19**(9) (2019)
13. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Proceedings of the International Conference on Neural Information Processing Systems. pp. 950–957 (1991)

14. Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., Thrun, S.: Towards fully autonomous driving: Systems and algorithms. In: Proceedings of the IEEE Intelligent Vehicles Symposium. pp. 163–168 (2011)
15. Lin, T.Y., Dollr, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 936–944 (2017)
16. Munir, F., Azam, S., Jeon, M., Lee, B.G., Pedrycz, W.: LDNet: End-to-end lane marking detection approach using a dynamic vision sensor. IEEE Transactions on Intelligent Transportation Systems pp. 1–17 (2021)
17. Nathan, S., Derek, H., Pushmeet, K., Rob, F.: Indoor segmentation and support inference from RGBD images. In: Proceedings of the European Conference on Computer Vision. pp. 746–760 (2012)
18. Nobis, F., Geisslinger, M., Weber, M., Betz, J., Lienkamp, M.: A deep learning-based radar and camera sensor fusion architecture for object detection. In: Proceedings of the Sensor Data Fusion: Trends, Solutions, Applications. pp. 1–7 (2019)
19. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035 (2019)
20. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. ArXiv (2021), `https://arxiv.org/abs/2103.13413`
21. Ravoor, P.C., Sudarshan, T.S.B.: Deep learning methods for multi-species animal re-identification and tracking a survey. Computer Science Review **38** (2020)
22. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 234–241 (2015)
23. Shekhar, H., Seal, S., Kedia, S., Guha, A.: Survey on applications of machine learning in the field of computer vision. In: Emerging Technology in Modelling and Graphics. pp. 667–678 (2020)
24. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014)
25. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the International Conference on Machine Learning. pp. 1139–1147 (2013)
26. Tao, A., Sapra, K., Catanzaro, B.: Hierarchical multi-scale attention for semantic segmentation. ArXiv (2020), `https://arxiv.org/abs/2005.10821`
27. Teichmann, M., Weber, M., Zollner, M., Cipolla, R., Urtasun, R.: MultiNet: Real-time joint semantic reasoning for autonomous driving. In: Proceedings of the IEEE Intelligent Vehicles Symposium. pp. 1013–1020 (2018)
28. Ye, J.C., Sung, W.K.: Understanding geometry of encoder-decoder CNNs. In: Proceedings of the International Conference on Machine Learning. pp. 7064–7073 (2019)
29. Yousefzadeh, A., Orchard, G., Gotarredona, T.S., Barranco, B.L.: Active perception with dynamic vision sensors. minimum saccades with optimum recognition. IEEE Transactions on Biomedical Circuits and Systems **12**(4), 927–939 (2018)
30. Zhang, Y., Yang, Q.: A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering (early access) (2021)