– Supplementary Material –
Multi-View Radar Semantic Segmentation

Arthur Ouaknine[1,2] Alasdair Newson[1] Patrick Pérez[2] Florence Tupin[1]
Julien Rebut[2]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
[2]valeo.ai, Paris, France
arthur.ouaknine@telecom-paris.fr

1. RAD tensor visualisation

An illustration of the RAD tensor is proposed in Figure 1. Each slice of 2D views corresponds to a discretized bin of the third axis. In Figure 1(b) for instance, the 256 range-Doppler slices correspond to the view of each discretized value of the angle axis. One can observe redundant signal information and a significant level of noise for each group of 2D-view slices.

2. Detailed multi-view architectures

We provide in Tables 1 and 2 the architecture details of the proposed multi-view network (MV-Net) and temporal multi-view network with ASPP modules (TMVA-Net) respectively. For each layer, the parameters of the operations used are specified in the following manner:

- n-dim convolution: Conv$n$D (input_channels, output_channels, kernel_size, stride, padding, dilation_rate);
- n-dim up-convolution: ConvTranspose$n$D (input_channels, output_channels, kernel_size, stride, padding, dilation_rate);
- maximum pooling: MaxPool2D (kernel_size, stride);
- atrous spatial pyramid pooling: ASPP (input_channels, output_channels);
- n-D batch normalisation: BNnD (input_channels);
- Leaky ReLU activation: LeakyReLU (negative_slope);

Where $n \in \{1, 2, 3\}$ is the dimension of the associated operation.

The ASPP module [1] has the same architecture as the one introduced by Kaul *et al*. [3] for range-angle semantic segmentation. We note that the 'output_channels' parameter for the ASPP module corresponds to the number of output channels for each parallel convolution. We also note that the 'stride' parameter can be either a scalar or a tuple of scalars depending on the axis on which it is applied.

3. Coherence loss

The purpose of the coherence loss (CoL) is to preserve a consistency between the predictions of the model for the different views of the same scene. The procedure used to construct this loss is illustrated in Figure 2.

4. Pre-processing and training procedures

The experiments in the main paper have been conducted using the parameters detailed in Table 3. An exponential decay with $\gamma = 0.9$ has been applied to each learning rate with an epoch step specific to each model (see Table 3). The competing methods have been trained using the Cross Entropy (CE) loss, except for the RSS-Net, which is trained with a weighted Cross Entropy (wCE) using the formulation in [3]. Our methods have been trained with the proposed combination of losses using the following parameters set up empirically: $\lambda_{\text{wCE}} = 1$, $\lambda_{\text{SDice}} = 10$ and $\lambda_{\text{CoL}} = 5$.

The architectures with which we compare our work have been designed to process inputs of size 256×256. Since the size of the range-Doppler view is 256×64 in the CARRADA dataset [5], it is resized in the Doppler dimension to train these competing models. On the other hand, our proposed architectures are composed of down-sampling layers adapted to the size of the Doppler dimension, thus they do not require this pre-processing step. The range-angle view has a size of 256×256 and does not require a resizing in both cases. For all methods, we used vertical and horizontal flip as data augmentation to reduce over-fitting.

Each view is normalised between 0 and 1 using local batch statistics for the competing methods. Our normali-

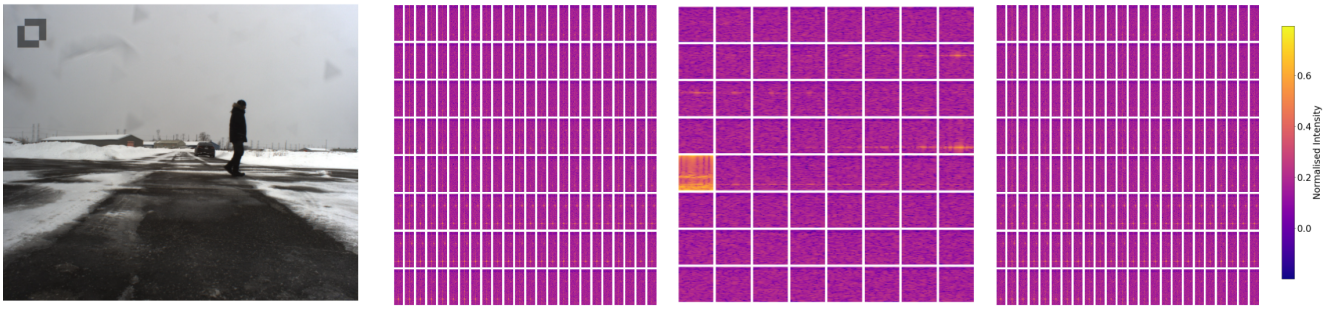

Figure 1: **Visualisation of the Range-Angle-Doppler (RAD) tensor.** (a) Camera image of the scene. The corresponding RAD tensor is visualised by slices of (b) range-Doppler, (c) range-angle or (d) angle-Doppler views w.r.t. their discretized third axis.

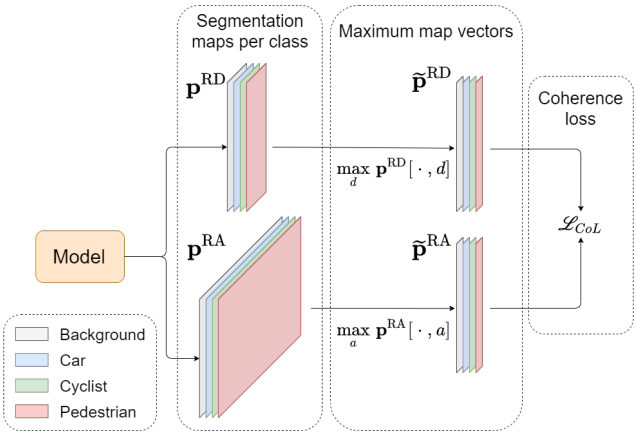

Figure 2: **Computation of the coherence loss.** The segmentation maps $\mathbf{p}^{\mathrm{RD}}$ and $\mathbf{p}^{\mathrm{RA}}$ of the two views are aggregated by max pooling along the axis that they do not share (either the Doppler or the angle). The coherence loss is the mean squared error (MSE) between the two resulting vectors $\tilde{\mathbf{p}}^{\mathrm{RD}}$ and $\tilde{\mathbf{p}}^{\mathrm{RA}}$.

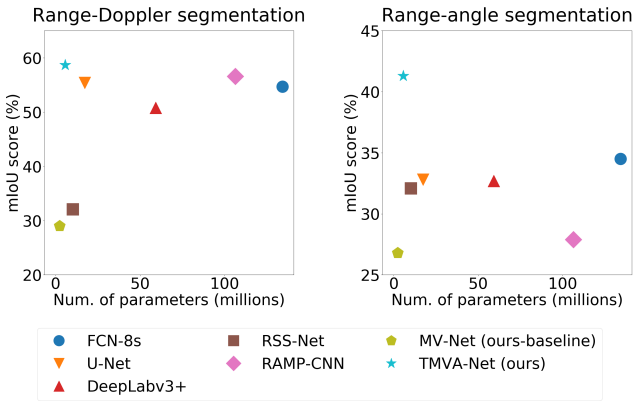

Figure 3: **Performance-*vs*.-complexity plots for all methods in RD and RA tasks.** Performance is assessed by mIoU (%) and complexity by the number of parameters (in millions) *for a single task*. Top-left models correspond to the best performing and the lightest. Only our models, MV-Net and TMVA-Net, are able to segment both views simultaneously. For all the other methods, two distinct models must be trained to address both tasks, which doubles the number of actual parameters.

sation strategy consists in using the global statistics of the entire CARRADA dataset to normalise the input views.

5. Quantitative results

Our proposed TMVA-Net architecture provides the best trade-off between performance and number of parameters for both range-Doppler and range-angle semantic segmentation tasks, as illustrated in Figure 3 with mIoU metric.

6. Qualitative results

Additional qualitative results are shown in Figure 4 for each method on scenes (1-2) from the CARRADA-Test. For the scene (1), RAMP-CNN (g) and TMVA-Net (i-j) display well segmented RD views. However, only TMVA-Net with CoL (j) is able to localise and classify both objects in the RD and RA views of the first example. In scene (2), four methods (d-e-i-j) are able to well localise objects in the RD view. Once again, only TVMA-Net with CoL (j) is able

	Layer	Inputs	Output resolution $(C \times H \times W)$	Operations
RD Encoder	rd_layer1	RD view	$128 \times 256 \times 64$	Conv2D(3, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer2	rd_layer1	$128 \times 128 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer3	rd_layer2	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer4	rd_layer3	$128 \times 64 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer5	rd_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
RA Encoder	ra_layer1	RA view	$128 \times 256 \times 256$	Conv2D(3, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer2	ra_layer1	$128 \times 128 \times 128$	MaxPool2D(2, 2)
	ra_layer3	ra_layer2	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer4	ra_layer3	$128 \times 64 \times 64$	MaxPool2D(2, 2)
	ra_layer5	ra_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
Latent space	layer6	rd_layer5, ra_layer5	$256 \times 64 \times 64$	concatenate(rd_layer5, ra_layer5)
RD Decoder	rd_layer7	layer6	$128 \times 64 \times 64$	Conv1D(256, 128, 1×1, 1, 0, 1)
	rd_layer8	rd_layer7	$128 \times 128 \times 64$	ConvTranspose2D(128, 128, 2×1, (2, 1), 1, 1)
	rd_layer9	rd_layer8	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer10	rd_layer9	$128 \times 256 \times 64$	ConvTranspose2D(128, 128, 2×1, (2, 1), 1, 1)
	rd_layer11	rd_layer10	$128 \times 256 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer12	rd_layer11	$K \times 256 \times 64$	Conv1D(128, K, 1×1, 1, 0, 1)
RA Decoder	ra_layer7	layer6	$128 \times 64 \times 64$	Conv1D(256, 128, 1×1, 1, 0, 1)
	ra_layer8	ra_layer7	$128 \times 128 \times 128$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer9	ra_layer8	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer10	ra_layer9	$128 \times 256 \times 256$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer11	ra_layer10	$128 \times 256 \times 256$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer12	ra_layer11	$K \times 256 \times 256$	Conv1D(128, K, 1×1, 1, 0, 1)

Table 1: **Multi-view network (MV-Net) architecture.** This table lists all the layers contained in the model taking as input multi-view radar representations (RD and RA views) to predict segmentation maps for each multi-view output. Details about the parameters of each operation are provided in Section 2. We note K the number of classes. The number of input channels in the first layer corresponds to the consecutive frames of each view stacked in temporal dimension, here $q = 2$ and thus the number of channels is 3.

to well segment objects in both RD and RA views while our method without CoL (i) predicts pedestrian and cyclist categories for pixels of the same object.

Figure 5 shows qualitative results for each method trained on CARRADA-Train and CARRADA-Val, and tested on in-house sequences of complex urban scenes (1-2) with a different range resolution. The qualitative examples and results have been cropped with respect to the minimum and maximum range of the dataset used for training. The ground-truth masks in columns (1-b) and (2-b) are empty because the radar views are not annotated. In scene (1), only TMVA-Net models (i-j) are able to localise and classify the signals related to the pedestrians and cars in both the RD and the RA views. In scene (2), only TMVA-Net (i-j) methods succeed to localise and classify cars and pedestrians in the RA view. We note that TMVA-Net without CoL (i) detects more car signals while TMVA-Net with CoL (j) is the only method capable of distinguishing pedestrian signatures on both RD and RA views.

These two examples in complex urban scenes suggest that our method has successfully learnt object signatures in the CARRADA dataset and is able to generalise well.

	Layer	Inputs	Output resolution ($C \times H \times W$)	Operations
RD Encoder	rd_layer1	RD view	$128 \times 256 \times 64$	Conv3D(1, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01) Conv3D(128, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01)
	rd_layer2	rd_layer1	$128 \times 128 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer3	rd_layer2	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer4	rd_layer3	$128 \times 64 \times 64$	MaxPool2D(2, (2, 1))
	rd_layer5	rd_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
	rd_layer6	rd_layer5	$640 \times 64 \times 64$	ASPP(128, 128)
	rd_layer7	rd_layer6	$128 \times 64 \times 64$	Conv1D(640, 128, 1×1, 1, 0, 1)
AD Encoder	ad_layer1	AD view	$128 \times 256 \times 64$	Conv3D(1, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01) Conv3D(128, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01)
	ad_layer2	ad_layer1	$128 \times 128 \times 64$	MaxPool2D(2, (2, 1))
	ad_layer3	ad_layer2	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ad_layer4	ad_layer3	$128 \times 64 \times 64$	MaxPool2D(2, (2, 1))
	ad_layer5	ad_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
	ad_layer6	ad_layer5	$640 \times 64 \times 64$	ASPP(128, 128)
	ad_layer7	ad_layer6	$128 \times 64 \times 64$	Conv1D(640, 128, 1×1, 1, 0, 1)
RA Encoder	ra_layer1	RA view	$128 \times 256 \times 256$	Conv3D(1, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01) Conv3D(128, 128, $3 \times 3 \times 3$, 1, (0, 1, 1), 1) + BN3D(128) + LeakyReLU(0.01)
	ra_layer2	ra_layer1	$128 \times 128 \times 128$	MaxPool2D(2, 2)
	ra_layer3	ra_layer2	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer4	ra_layer3	$128 \times 64 \times 64$	MaxPool2D(2, 2)
	ra_layer5	ra_layer4	$128 \times 64 \times 64$	Conv1D(128, 128, 1×1, 1, 0, 1)
	ra_layer6	ra_layer5	$640 \times 64 \times 64$	ASPP(128, 128)
	ra_layer7	ra_layer6	$128 \times 64 \times 64$	Conv1D(640, 128, 1×1, 1, 0, 1)
Latent space	layer8	rd_layer5, ra_layer5, ad_layer5	$384 \times 64 \times 64$	concatenate(rd_layer5, ra_layer5, ad_layer5)
RD Decoder	rd_layer9	layer8	$128 \times 64 \times 64$	Conv1D(384, 128, 1×1, 1, 0, 1)
	rd_layer10	rd_layer7, rd_layer9, ad_layer7	$384 \times 64 \times 64$	concatenate(rd_layer7, rd_layer9, ad_layer7)
	rd_layer11	rd_layer10	$128 \times 128 \times 64$	ConvTranspose2D(384, 128, 2×1, (2, 1), 1, 1)
	rd_layer12	rd_layer11	$128 \times 128 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer13	rd_layer12	$128 \times 256 \times 64$	ConvTranspose2D(128, 128, 2×1, (2, 1), 1, 1)
	rd_layer14	rd_layer13	$128 \times 256 \times 64$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	rd_layer15	rd_layer14	$K \times 256 \times 64$	Conv1D(128, K, 1×1, 1, 0, 1)
RA Decoder	ra_layer9	layer8	$128 \times 64 \times 64$	Conv1D(384, 128, 1×1, 1, 0, 1)
	ra_layer10	ra_layer7, ra_layer9, ad_layer7	$384 \times 64 \times 64$	concatenate(ra_layer7, ra_layer9, ad_layer7)
	ra_layer11	ra_layer10	$384 \times 128 \times 128$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer12	ra_layer11	$128 \times 128 \times 128$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer13	ra_layer12	$128 \times 256 \times 256$	ConvTranspose2D(128, 128, 2×2, 2, 1, 1)
	ra_layer14	ra_layer13	$128 \times 256 \times 256$	Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01) Conv2D(128, 128, 3×3, 1, 1, 1) + BN2D(128) + LeakyReLU(0.01)
	ra_layer15	ra_layer14	$K \times 256 \times 256$	Conv1D(128, K, 1×1, 1, 0, 1)

Table 2: **Temporal multi-view network with ASPP modules (TMVA-Net) architecture.** This table lists all the layers contained in the model taking as input multi-view radar representations (RD and RA views) to predict segmentation maps for each multi-view output. Details about the parameters of each operation are provided in Sec. 2. We note K the number of classes. The number of input channels in the first layer is fixed to 1 because the consecutive frames are considered as a sequence, here $q = 4$ and thus the number of channels is 5.

View	Method	Param. #	q	Batch size	LR	LR step	Epoch #
RD	FCN-8s [4]	134.3	0	20	10^{-4}	10	100
	U-Net [6]	17.3	3	6	10^{-4}	20	150
	DeepLabv3+ [2]	59.3	3	20	10^{-4}	20	150
	RSS-Net	10.1	3	6	10^{-3}	10	100
	RAMP-CNN	106.4	9	2	10^{-5}	20	150
	MV-Net (ours-baseline)	2.4*	3	13	10^{-4}	20	300
	TMVA-Net (ours)	5.6*	5	6	10^{-4}	20	300
RA	FCN-8s [4]	134.3	0	10	10^{-4}	10	100
	U-Net [6]	17.3	3	6	10^{-4}	20	150
	DeepLabv3+ [2]	59.3	3	20	10^{-4}	20	150
	RSS-Net	10.1	3	6	10^{-4}	10	100
	RAMP-CNN	106.4	9	2	10^{-5}	20	150
	MV-Net (ours-baseline)	2.4*	3	13	10^{-4}	20	300
	TMVA-Net (ours)	5.6*	5	6	10^{-4}	20	300

Table 3: **Hyper-parameters used for training.** The number of trainable parameters (in millions) for each method corresponds to a single view-segmentation model; Two such models, one for each view, are required for all methods but ours. In contrast, the number of parameters reported for our methods ('*') corresponds to a single model that segments both RD and RA views. RSS-Net and RAMP-CNN have been modified to be trained on both tasks (see Sec. 4.2 of the main article). The input of a model consists in $q+1$ successive RAD frames, where q is the number of considered past frames, if any. The learning rate ('LR') step is in epochs.

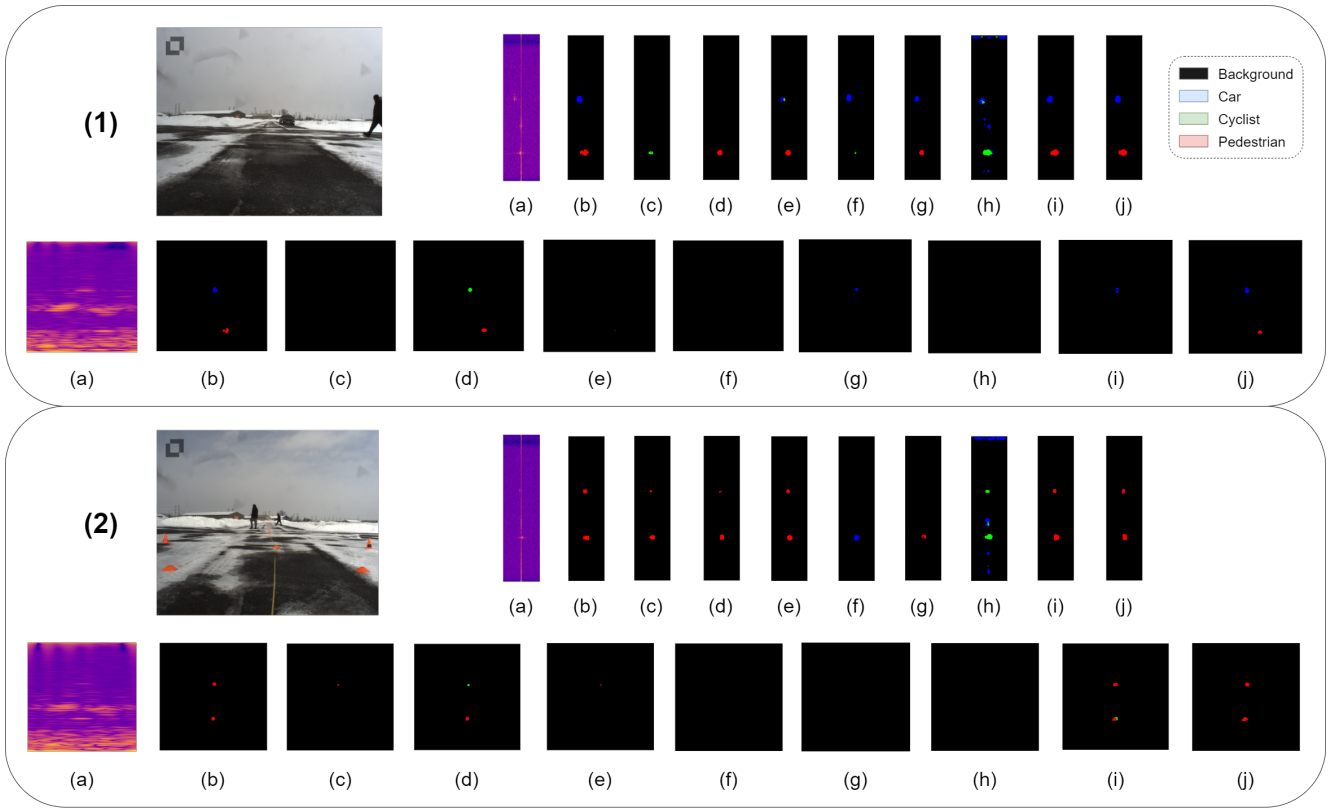

Figure 4: **Qualitative results on two test scenes of CARRADA-Test**. (1) and (2) are two independent examples. (*Top*) camera image of the scene and results of the RD segmentation; (*Bottom*) Results of the RA Segmentation. (a) Radar view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice loss), (i) TMVA-Net (ours, w/ wCE+SDice loss), (j) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. In *TPAMI*, 2017. 1

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 5

[3] Prannay Kaul, Daniele De Martini, Matthew Gadd, and Paul Newman. RSS-Net: weakly-supervised multi-class semantic segmentation with FMCW radar. In *IV*, 2020. 1

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*,

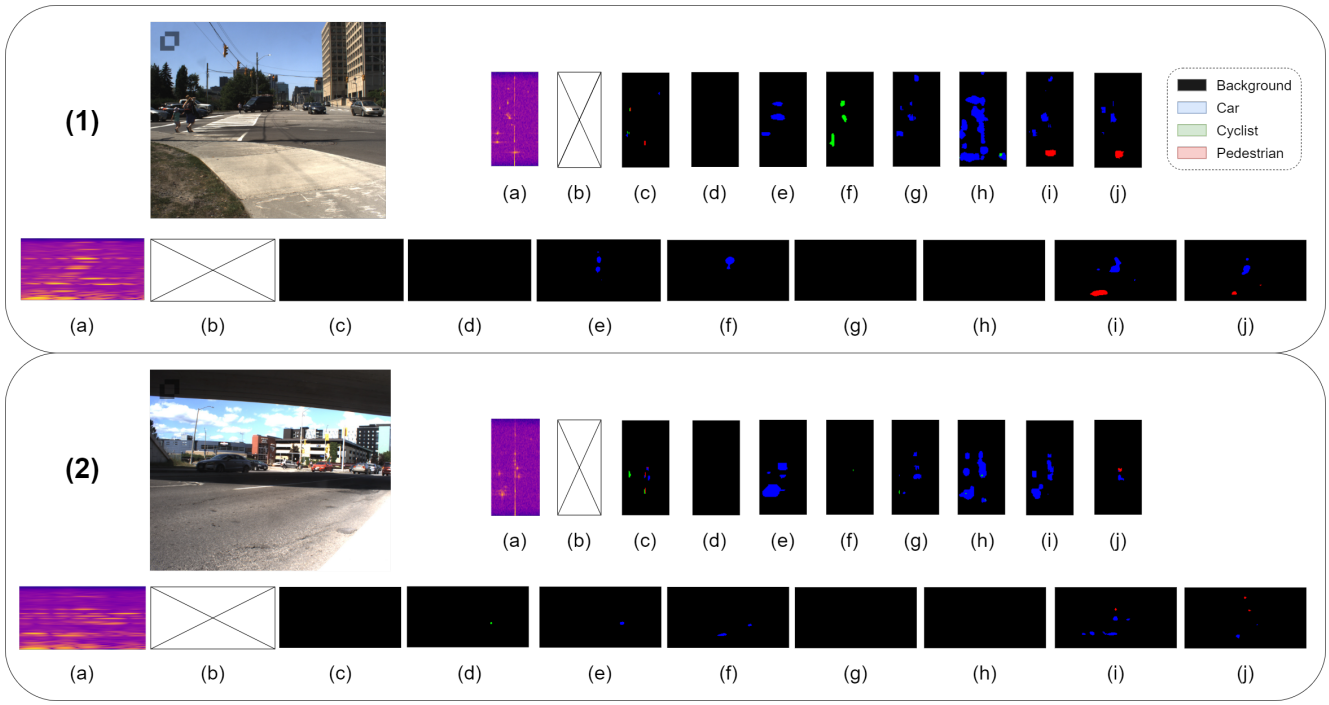

Figure 5: **Qualitative results on two test scenes of in-house sequences**. (1) and (2) are two independent examples. (*Top*) camera image of the scene and results of the RD segmentation; (*Bottom*) Results of the RA Segmentation. (a) Radar view signal, (b) ground-truth mask, (c) FCN8s, (d) U-Net, (e) DeepLabv3+, (f) RSS-Net, (g) RAMP-CNN, (h) MV-Net (our baseline w/ wCE+SDice loss), (i) TMVA-Net (ours, w/ wCE+SDice loss), (j) TMVA-Net (ours, w/ wCE+SDice+CoL loss).

2015. 5

[5] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez. CARRADA dataset: camera and automotive radar with range-angle-Doppler annotations. In *ICPR*, 2020. 1

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5