Multi-Domain Incremental Learning for Semantic Segmentation Supplementary Material

Prachi Garg[1] Rohit Saluja[1] Vineeth N Balasubramanian[2] Chetan Arora[3]
Anbumani Subramanian[1] C.V. Jawahar[1]
[1]CVIT - IIIT Hyderabad, India [2]IIT Hyderabad, India [3]IIT Delhi, India
[1]prachigarg2398@gmail.com, [1]rohit.saluja@research.iiit.ac.in, [2]vineethnb@iith.ac.in,
[3]chetan@cse.iitd.ac.in, [1]{anbumani, jawahar}@iiit.ac.in

In this supplementary material, we discuss more details of our implementation as well as additional results and analysis which could not be included in the main paper owing to space constraints.

A1. Ordering of domains

In this section, we explore more sequences of domain ordering. We show results on $BDD \rightarrow CS$ and $BDD \rightarrow CS \rightarrow IDD$ in Table A1, on $IDD \rightarrow CS$ and $IDD \rightarrow CS \rightarrow BDD$ in Table A2. It is evident that there is a significant drop in performance as the model is fine-tuned in a naive manner. Our method has been designed keeping in mind a good stability-plasticity trade-off, and our model performs better than the FE and FT baselines.

A2. Implementation Details

A2.1. Architecture

We perform all experiments on the ERFNet [5] architecture as it allows the dynamic addition of our modules seamlessly. This is shown in Figure A2. It uses a modified ResNet such that the standard 3×3 convolutional layer is factorized into a stack of one 3×1 and one 1×3 convolutional layers. The convolutional layer before the first residual block in the encoder is domain-shared. We use a DS-BN layer before the first residual block in the encoder. The DAU units are applied across the entire depth of the network. Input image resolution for all datasets is resized to 1024×512. Training on all experiments is performed using a batch size of 6, momentum of 0.9 and weight decay of 1e$-$4 for 150 epochs.

A2.2. Training Details

For our proposed model, we train the domain-specific weights W_t using the standard learning rate of 5e-4 and shared weights W_s using a *dlr* learning rate that is 100x lower, i.e., 5e-6. After hyperparameter tuning, we find that a regularization factor of $\lambda_{KLD} = 0.1$ works best. We compute class balancing weights for each dataset and use the respective weights for evaluation and validation. We now discuss training protocols used for our baselines and comparisons.

Fine-tuning. (Section 4 in the main paper) This acts as a standard IL baseline and forms our lower bound (Tables 1 and 2). To train the fine-tuning baseline, we add a randomly initialized decoder head and fine-tune the shared encoder and new decoder on the new dataset.

Multi-task (Joint Training). This baseline (Section 4, Tables 1 and 2 of main paper) helps us compare our incremental settings against offline training. We train this model simultaneously on all the domains. We have a single shared encoder (with default ERFNet backbone) and multiple decoder heads, one for each domain. Training is done on the datasets in alternate batches. Gradients in the shared encoder are updated for all batches, while gradients in decoder are updated only on batches of their respective domains. Learning rate for shared weights is the standard learning rate divided by the number of domains.

RAP-FT model in Step 1. To train our proposed model in step 1, we attach the domain-specific RAPs and domain-specific BN layers to standard ERFNet and fine-tune all model parameters (W_s, W_t) at the same standard learning rate of 5e-6. The W_ts were randomly initialized while the W_s-es were initialized from an Imagenet pre-trained encoder. This refers to the 71.82% value in Table 1 of main paper.

Training protocol used for comparison with existing residual adapters (Table 6 of main paper). We compare our method against the parallel residual adapter (RAP-I), series residual adapters (RAS-I) and reparametrized convolutions for multi-task learning (RCM-I) methods. In all these comparisons, the shared weights are frozen to Imagenet pre-training. Since the shared weights are not trainable in any of these models, mIoU on all previous domains remains con-

IL Step	Step 1 BDD		Step 2: $D_A \neq D_B, \mathcal{Y}_A \neq \mathcal{Y}_B$ $BDD \rightarrow CS$			Step 3 $BDD \rightarrow CS \rightarrow IDD$			
Methods	BDD ↑	$\Delta_m\%$ ↓	BDD ↑	CS ↑	$\Delta_m\%$ ↓	BDD ↑	CS ↑	IDD ↑	$\Delta_m\%$ ↓
Single-task	54.1		54.1	72.55		54.1	72.55	61.97	
Multi-task	54.1		57.69	69.42	1.16% (↑)	58.13	69.37	59.37	0.38%
FT	54.1	0.0	25.4	70.43	27.99%	24.15	40.73	60.8	33.70%
FE	54.1	0.0	54.1	55.75	11.58%	54.1	55.75	46.56	16.01%
Ours	52.1	3.70	46.03	67.20	**11.14**	49.3	59.17	56.1	**12.26%**

Table A1: $BDD \rightarrow CS \rightarrow IDD$

IL Step	Step 1 IDD		Step 2: $D_A \neq D_B, \mathcal{Y}_A \neq \mathcal{Y}_B$ $IDD \rightarrow CS$			Step 3 $IDD \rightarrow CS \rightarrow BDD$			
Methods	IDD ↑	$\Delta_m\%$ ↓	IDD ↑	CS ↑	$\Delta_m\%$ ↓	IDD ↑	CS ↑	BDD ↑	$\Delta_m\%$ ↓
Single-task	61.97		61.97	72.55		61.97	72.55	54.1	
Multi-task	61.97		60.85	71.11	1.90%	59.37	69.37	58.13	0.38%
FT	61.97	0.0	23.78	71.18	31.76%	25.65	44.77	53.72	32.53%
FE	61.97	0.0	61.97	58.25	9.86%	61.97	58.25	44.96	12.20%
Ours	62.60	1.02 (↑)	55.91	70.34	**6.41%**	52.24	60.39	51.03	**12.71%**

Table A2: $IDD \rightarrow CS \rightarrow BDD$

stant and the models can be trained independently of each other, unlike our method where each subsequent IL step is to be initialized from the previous IL step. 1×1 convolutional domain-specific adapter layers are added in series with existing residual layers, at each layer for RAS-I and at each block for RCM-I model. We use random initialization for these RAS and RCM adapter layers while the W_s parameters are frozen to Imagenet pre-trained initialization. The topology of RAP, RAS and RCM blocks in our experiments is illustrated in A3.

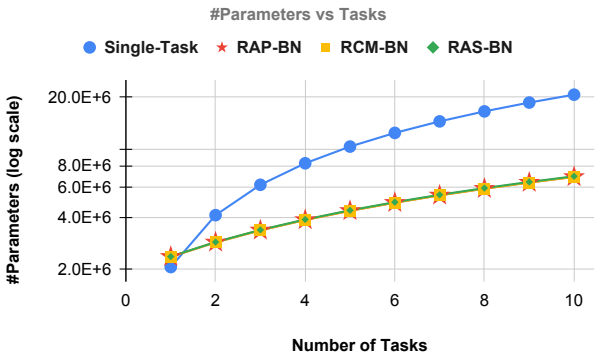


Figure A1: Parameter growth analysis in our experiments. The RAP, RAS and RCM residual adapters have the same percentage of parameter growth (21.2%).

A3. Choice of Residual Adapter

In this section, we explain why we chose the parallel residual adapter (RAP) from [4] instead of the series residual adapter (RAS) proposed in [3] or the reparametrized convolution module (RCM) formalized in [2]. These works originally proposed to train the domain-specific adapters on each new domain in IL step t, while the shared convolutions W_s remain frozen to an Imagenet pre-trained initialization. In Table A3, we show experiments on the different types of residual adapters, RCM, RAS and RAP trained using our proposed optimization strategy.

In each of these models, we train both W_s and W_t only on the domain-specific loss L_{CE_t} using our $init_{W_t}$ and *dlr* optimization strategy rather than freezing the shared weights W_s. It can be seen that the parallel residual adapter (RAP) is the best suited to our optimization and we select this adapter over RCM and RAS. As it is added to an existing residual unit's layer in parallel, RAP adapters can easily be added to off-the-shelf segmentation models with a ResNet based backbone. For fair comparison, we use DS-BN layers in all these models.

A4. Parameter Growth Analysis

In Figure A1, we show the increase in number of parameters with each incremental task for the different models. The single-task model has a 100% growth in parameters. Our model outperforms baselines and comparative methods with only a 21.2% growth in parameters at each incremental step.

IL Step	Step 1		Step 2: $D_A \neq D_B, \mathcal{Y}_A = \mathcal{Y}_B$			Step 2: $D_A \neq D_B, \mathcal{Y}_A \neq \mathcal{Y}_B$			Step 3: $D_A \neq D_B, \mathcal{Y}_A \neq \mathcal{Y}_B$			
	CS		$CS \rightarrow BDD$			$CS \rightarrow IDD$			$CS \rightarrow BDD \rightarrow IDD$			
Methods	CS ↑	$\Delta_m\% \downarrow$	CS ↑	BDD ↑	$\Delta_m\% \downarrow$	CS ↑	IDD ↑	$\Delta_m\% \downarrow$	CS ↑	BDD ↑	IDD ↑	$\Delta_m\% \downarrow$
Single-task	72.55	NA	72.55	54.1	NA	72.55	61.97	NA	72.55	54.1	61.97	NA
RCM-FT-dlr	68.04	6.22%	55.88	53.76	11.80%	50.38	55.17	20.77%	46.84	37.78	55.54	25.33%
RAS-FT-dlr	71.19	1.87%	48.89	56.12	14.44%	43.13	57.7	23.72%	33.09	37.6	58.56	30.13%
RAP-FT-dlr (ours)	71.82	1.01%	**57.30**	**57.15**	**7.69%**	**57.39**	**59.72**	**12.26%**	**50.79**	**48.86**	**58.9**	**14.88%**

Table A3: Comparison of RAP, RAS and RCM models where shared weights W_s are fine-tuned using our proposed optimization strategy.

		Methods	*Road*	*Sidewalk*	*Building*	*Wall*	*Fence*	*Pole*	*T-light*	*T-sign*	*Vegetation*	*Terrain*	*Sky*	*Person*	*Rider*	*Car*	*Truck*	*Bus*	*Train*	*Motorcycle*	*Bicycle*	mIoU (↑)	$\Delta_m\%(\downarrow)$
		Single-Task	97.6	81.7	91.0	53.3	55.6	61.1	62.3	71.6	91.3	62.6	93.3	75.7	54.8	93.0	69.9	79.7	65.3	48.5	70.1	72.55	
CS→BDD	Cityscapes	FT	92.3	58.8	82.7	26.2	22.9	43.6	0.1	37.8	84.7	37.5	75.4	33.4	5.5	86.6	31.1	26.3	13.0	2.2	0.6	40.05	23.66
		RAP-I [4]	96.5	74.4	87.1	40.2	43.3	47.7	**51.0**	61.7	**89.1**	**54.4**	**89.5**	65.8	37.5	88.9	30.7	43.7	15.2	30.0	**63.9**	58.43	16.90
		Ours	**96.9**	**77.6**	**88.3**	**45.0**	**48.2**	**56.4**	48.7	**62.3**	88.4	53.9	88.0	**68.8**	**45.7**	**91.1**	**62.6**	**64.5**	**59.8**	**30.6**	62.4	**65.21**	**3.55**
CS→IDD	Cityscapes	FT	86.9	45.4	76.0	18.0	14.3	37.4	0.0	19.5	83.7	31.9	41.2	56.9	16.1	79.5	14.4	25.0	4.1	12.1	36.9	36.81	24.96
		RAP-I [4]	96.5	74.4	87.1	**40.2**	43.3	47.7	**51.0**	**61.7**	89.1	**54.4**	**89.5**	65.8	37.5	88.9	30.7	43.7	15.2	30.0	63.9	58.43	18.61
		Ours	**96.6**	**77.5**	**88.4**	40.1	**47.0**	**56.3**	48.6	61.2	**89.5**	48.2	86.1	**70.4**	**47.6**	**90.6**	**57.6**	**64.0**	**60.1**	**32.9**	**64.4**	**64.58**	**7.80**
CS→BDD→IDD	Cityscapes	FT	85.93	46.5	70.3	20.5	15.9	25.4	0.0	20.3	82.9	33.6	17.9	40.8	1.4	76.1	8.7	21.3	5.5	0.25	5.8	30.49	33.62
		RAP-I [4]	**96.5**	74.4	87.1	**40.2**	**43.3**	47.7	**51.0**	**61.7**	**89.1**	**54.4**	**89.5**	65.8	37.5	88.9	30.7	43.7	15.2	**30.0**	**63.9**	58.43	17.02
		Ours	96.2	**74.5**	**87.4**	30.4	41.6	**50.4**	32.9	49.2	87.9	47.1	85.3	**67.9**	**38.9**	**90.5**	**54.2**	**64.9**	**52.2**	12.6	60.6	**59.19**	**10.39**
	BDD	FT	69.4	24.3	67.2	8.3	19.7	34.5	0.3	23.6	78.1	33.7	82.9	41.9	0.02	59.2	13.9	36.9	0.0	2.4	12.8	32.05	33.62
		RAP-I [4]	91.8	52.8	80.6	20.4	32.9	42.8	**46.8**	**42.9**	82.4	**43.3**	**93.3**	53.2	0.0	**86.0**	33.1	45.1	0.0	**21.9**	11.1	46.34	17.02
		Ours	**92.5**	**59.4**	**82.7**	**22.2**	**40.0**	**48.0**	32.9	39.2	**83.0**	43.1	**93.3**	**57.4**	**18.8**	85.3	**40.5**	**51.0**	**0.0**	19.5	**34.7**	**49.66**	**10.39**

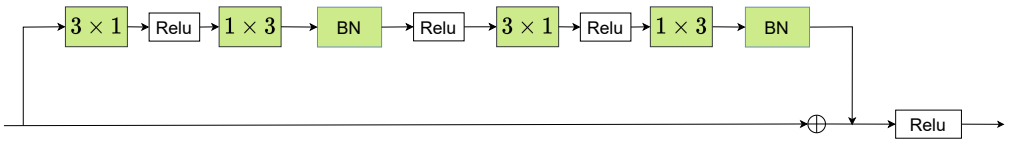
Table A4: Numerical evaluation of catastrophic forgetting in our incremental learning settings in terms of class-wise IoU. We show evaluation only on the validation sets of previous domains after incrementally learning on the subsequent domains. Thus, in $CS \rightarrow BDD$ setting, we show forgetting on CS after learning on BDD. In $CS \rightarrow IDD$, we show performance on CS after learning on IDD. In $CS \rightarrow BDD \rightarrow IDD$, we show performance on both CS and BDD after learning on IDD. While mIoU is computed on each domain, $\Delta_m\%$ is the overall IL performance score computed on all the domains the model has learned so far.

A5. Class-wise Accuracy and Qualitative Analysis

In continuation to $CS \rightarrow BDD$ results, section 4 of main paper, we present detailed class-wise performance analysis. Classes like traffic light and traffic sign are visually different in German (CS) and American (BDD) roads due to infrastructural differences in the appearance and location of these objects [6]. The fine-tuning model completely forgets on traffic lights, Figure 4 (a) main paper. Our model mitigates forgetting in all 19 classes, and retains performance by a significantly large margin ($\geq 30\%$) on *safety-critical classes* such as traffic light, traffic sign, person, rider, truck, bus and bicycle. Importantly, we observe that our proposed model has surpassed the single-task model performance on BDD by 1.63%. We hypothesize that this forward transfer is achieved since our model captures the domain-specific characteristics of the dataset distributions of CS and BDD in the domain-specific parameters. Since the label spaces of the two domains are overlapping, domain interference is thus avoided. This helps in achieving forward transfer wherein BDD has benefited from the knowledge already learned for CS in the previous step. Classes like rider, bicycle, motorcycle and train have only a few hundred instance occurrences in the entire BDD dataset as compared to other classes like cars and poles, which occur at an average of 10 instances per image ([6]). As such, the BDD dataset doesn't have enough instances to train well on these classes. The CS model when fine-tuned on BDD hence performs poorly on rider, motorcycle and bicycle (Figure 4(b), main paper). Our proposed method outperforms even the single-task baseline on these classes: person (1.36%), rider (16.52%), motorcycle (4.78%) and bi-

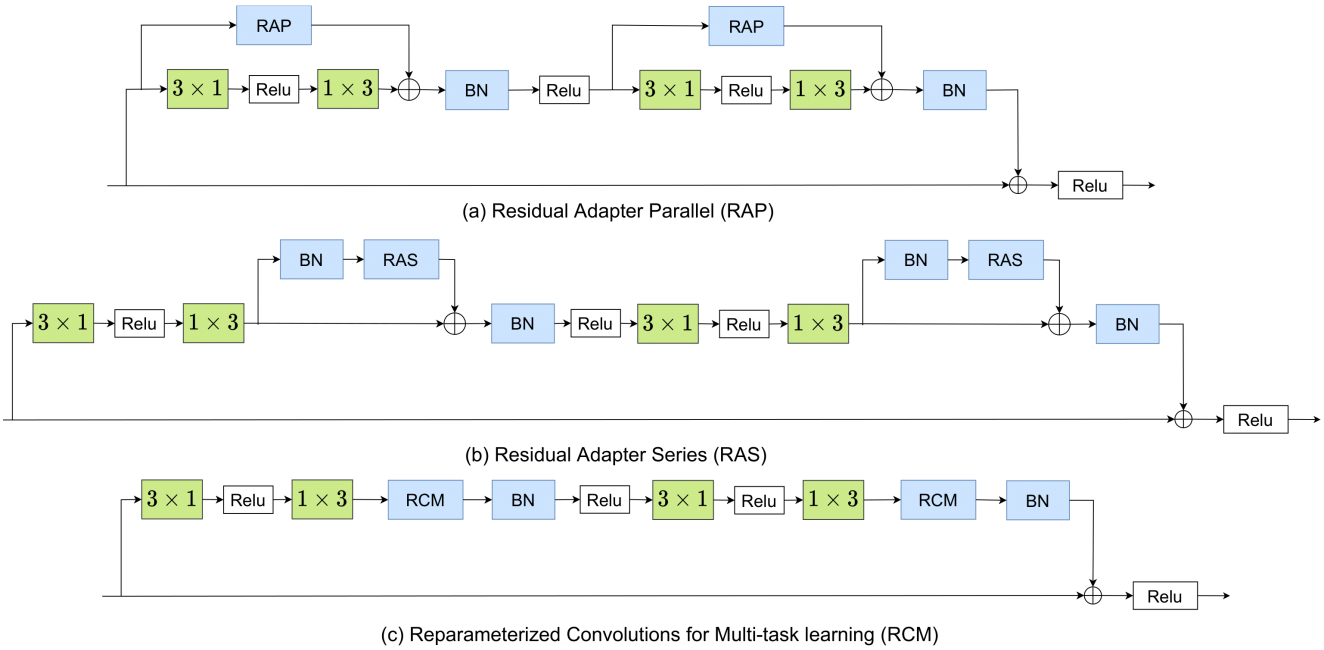
cycle (14.93%). These observations show that our model has achieved a *forward knowledge transfer* for these classes without having access to the old dataset.

We show more qualitative analysis in the form of T-SNE plots in Figure A7. Table A4 shows the class-wise evaluation of our proposed method, compared with the fine-tuning baseline and the RAP-I method which uses shared weights frozen from a model pre-trained on Imagenet. Note that the RAP-I model has frozen shared weights and domain-specific weights of previous domains were also frozen. Hence, there is no change in performance of any previously learned domain in RAP-I model. But it exhibits poor plasticity towards new domain (please see Table 6 in the main paper). The overall score $\Delta_m\%$ is indicative of the performance of the model on new as well as previous domains. Our model consistently improves performance on safety-critical classes like person, rider, car, truck, motorcycle, bicycle, bus and train by a large margin. Please see Figure A4, where we compare output segmentation maps from our proposed model with those of the fine-tuning and RAP-I-vanilla methods on CS dataset after incrementally learning $CS \rightarrow BDD$. It is evident that our results show significant improvement over the RAP-I-vanilla and fine-tuning methods and are closer to the ground truth maps. Our model is able to achieve a *forward knowledge transfer* from CS to BDD for the classes rider, bicycle, motorcycle, person (which were abundant in CS but have fewer instances in BDD). The above inference is validated in the output segmentation maps in Figure A5. In Figure A6, we show confusion matrices for the Cityscapes validation set after learning on BDD and IDD in step 2. It can be observed that our model is more confident in its prediction of safety-critical classes as compared to the fine-tuning baseline for the same.



(a) Original residual unit in ERFNet

Figure A2: Residual unit of the ERFNet encoder [5]. The 3×3 convolutional layer in the original ResNet [1] is replaced by a stack of 3×1 and 1×3 layers with a ReLU in between.



(a) Residual Adapter Parallel (RAP)

(b) Residual Adapter Series (RAS)

(c) Reparameterized Convolutions for Multi-task learning (RCM)

Figure A3: Modified residual adapter units as used in our experiments in Sec 5. The green layers are the shared layers w_1, w_2 in our DAU. Domain-specific layers in blue. The RAP, RAS and RCM layers are all 1×1 convolutional layers. They are specific to their respective domains, we have shown domain-specific layers for a single domain for simplicity. (a) **RAP: [4]** This is detailed illustration of our DAU unit. (b) **RAS: [3]** This is the series residual adapter. (c) **RCM: [2]** This series residual adapter is designed for multi-task incremental learning on tasks like human parts detection, edge detection, etc. We compare our method against the RAS and RCM adapters. Domain-specific Batch Normalization is used in all three adapters for a fair comparison.

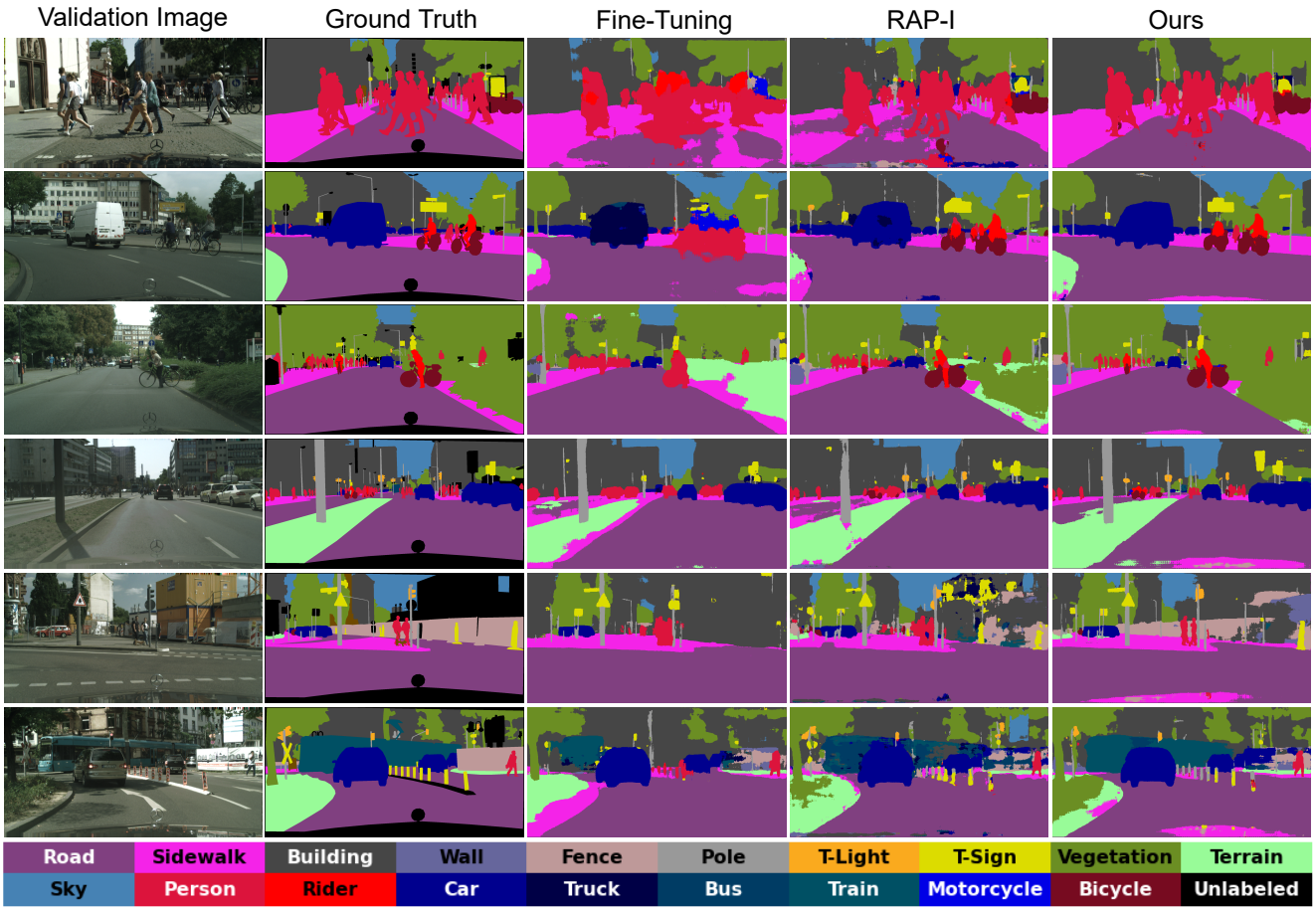


Figure A4: Semantic segmentation of sample scenes extracted from the **Cityscapes validation set** after the CS model has been incrementally trained on BDD dataset (after $CS \rightarrow BDD$). This shows the performance on previous domain after the subsequent incremental step. The fine-tuning model undergoes catastrophic forgetting and performs poorly. Our model is able to mitigate forgetting by a large extent as seen in these samples.

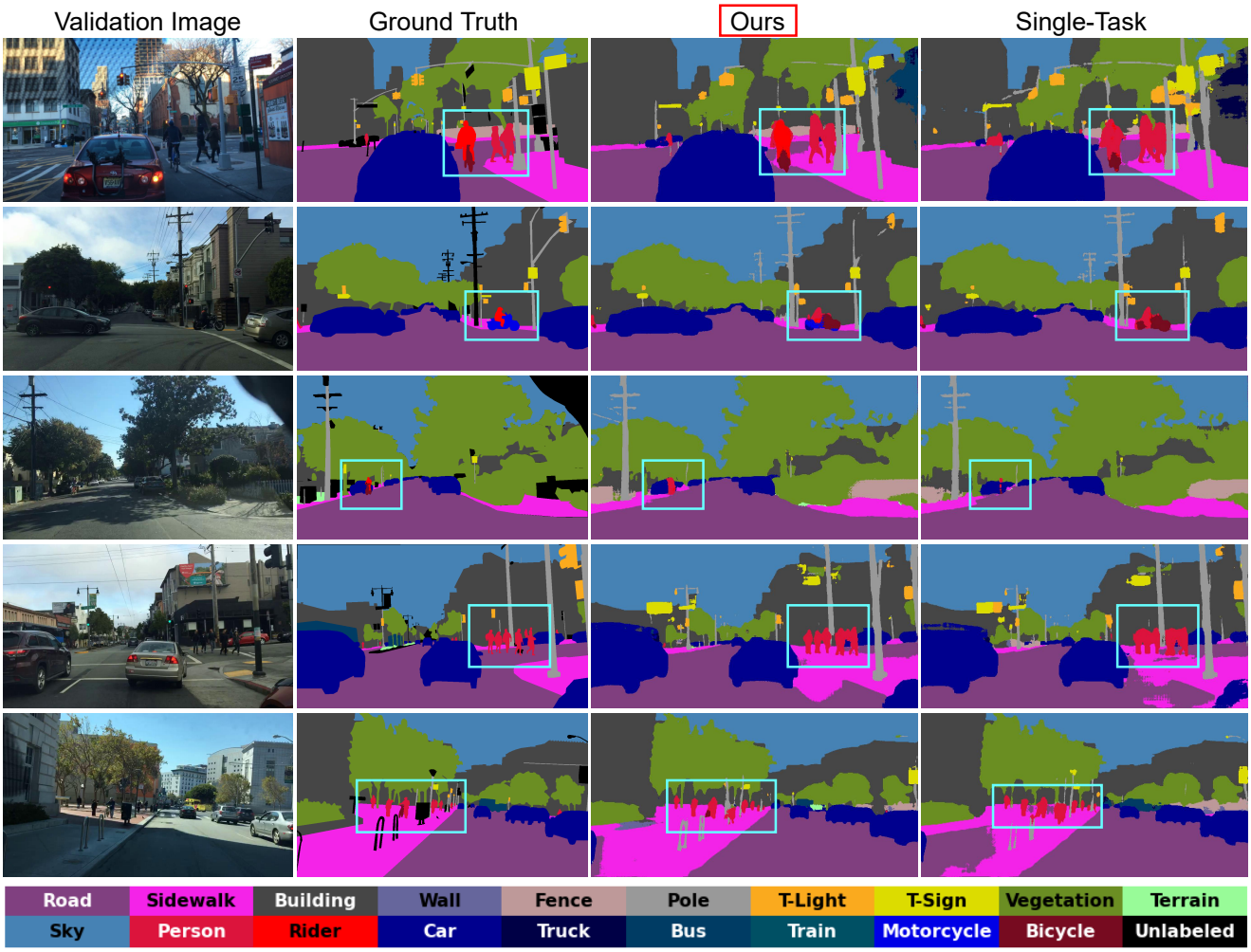
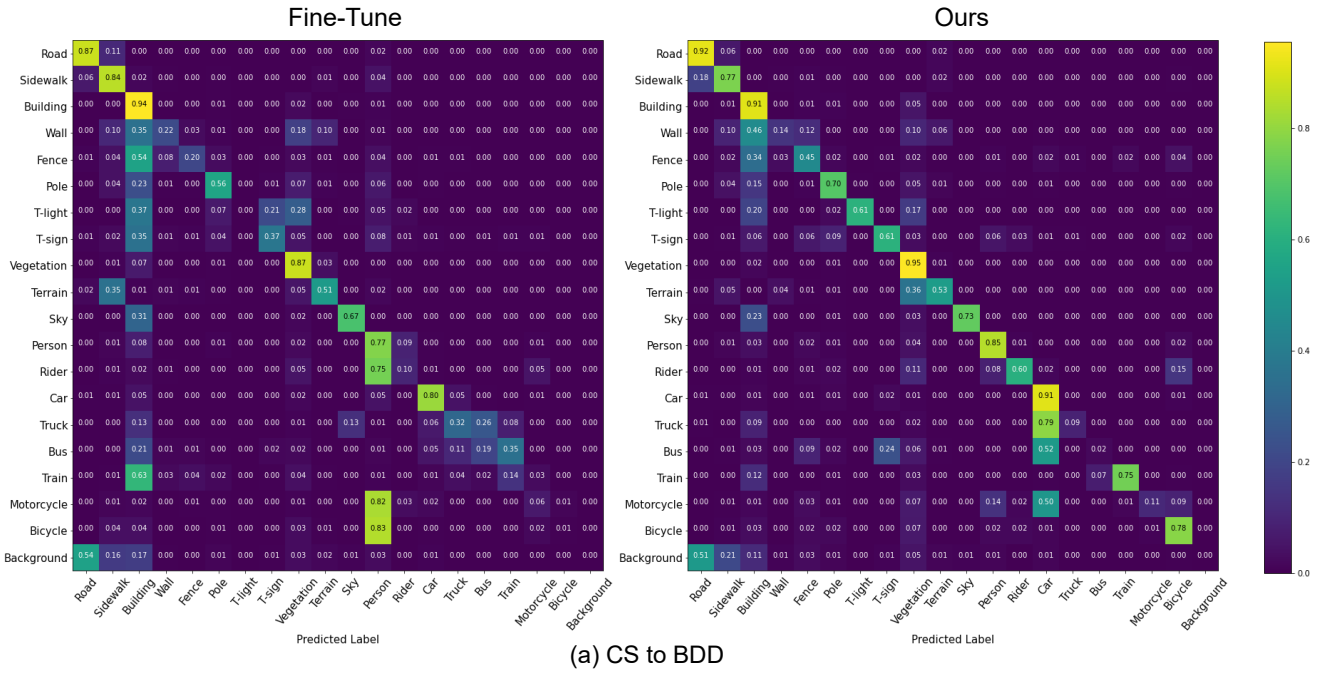
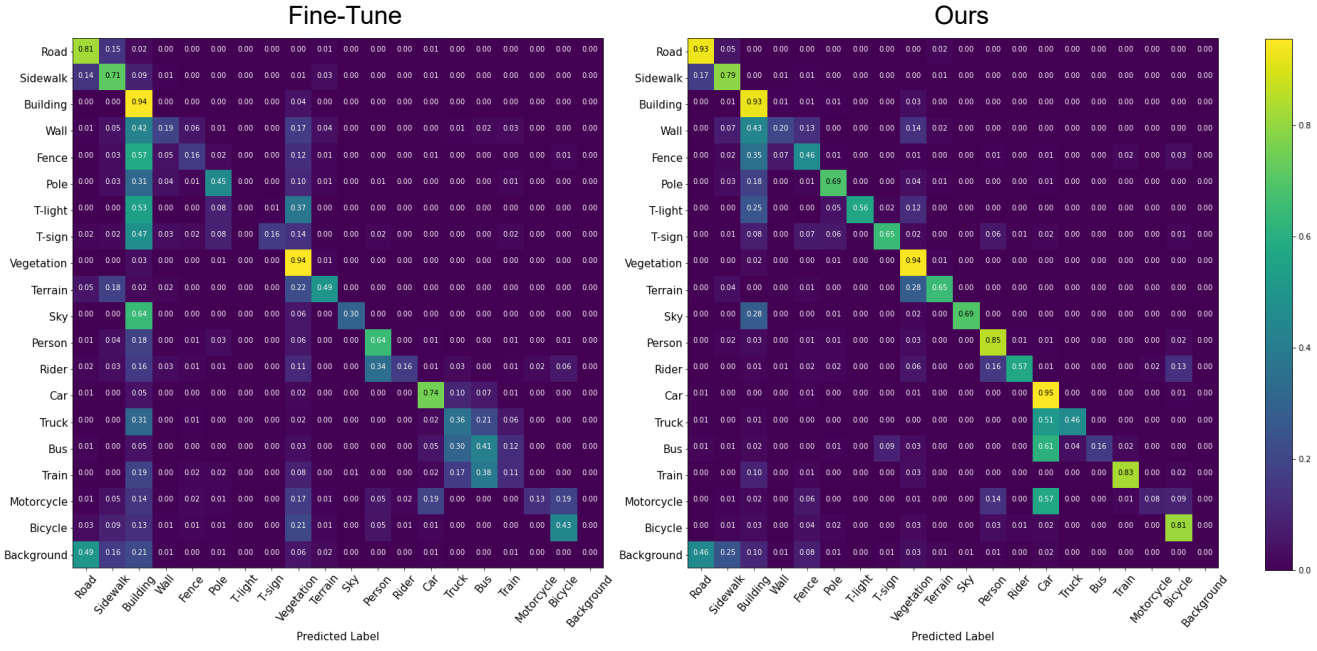


Figure A5: Semantic segmentation of sample scenes extracted from the **BDD validation set** after the CS model has been incrementally trained on BDD dataset (after $CS \rightarrow BDD$). Note that our model performs better than the single-task baseline for the classes **rider, person, motorcycle, bicycle**. This is called forward transfer in incremental learning. In the 1st row, our model is able to clearly identify the rider on top of a bicycle whereas, the single-task baseline marks that as a person. In the second row, the single-task model marks a motorcycle as bicycle while our model is able to correctly identify the motorcycle. Similarly, persons and riders are more clearly segmented by our model.

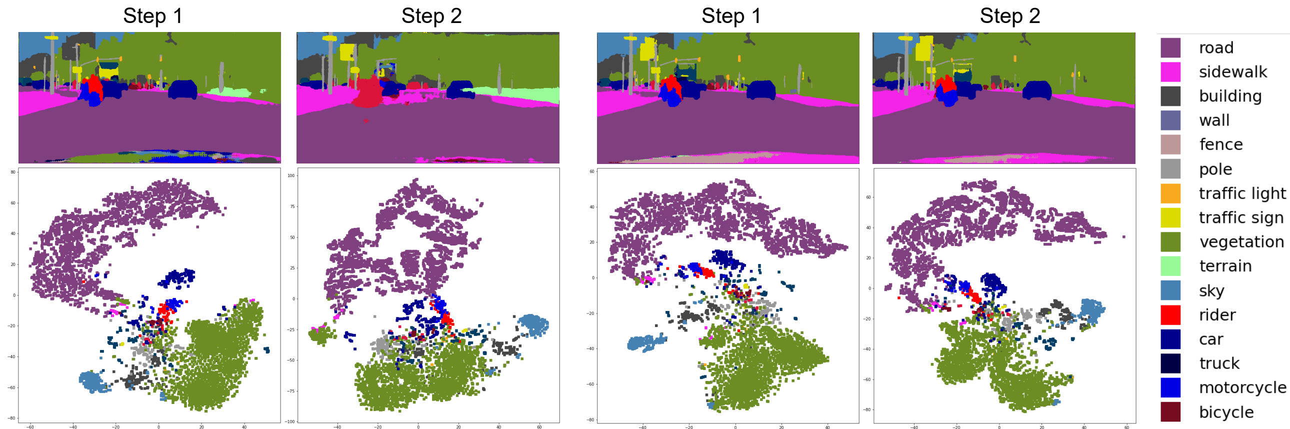


(a) CS to BDD

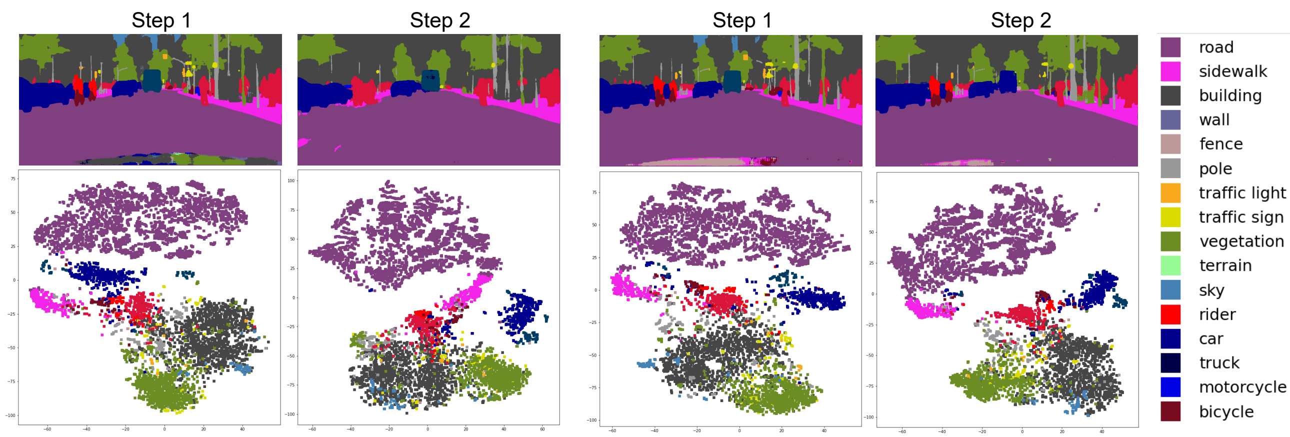


(b) CS to IDD

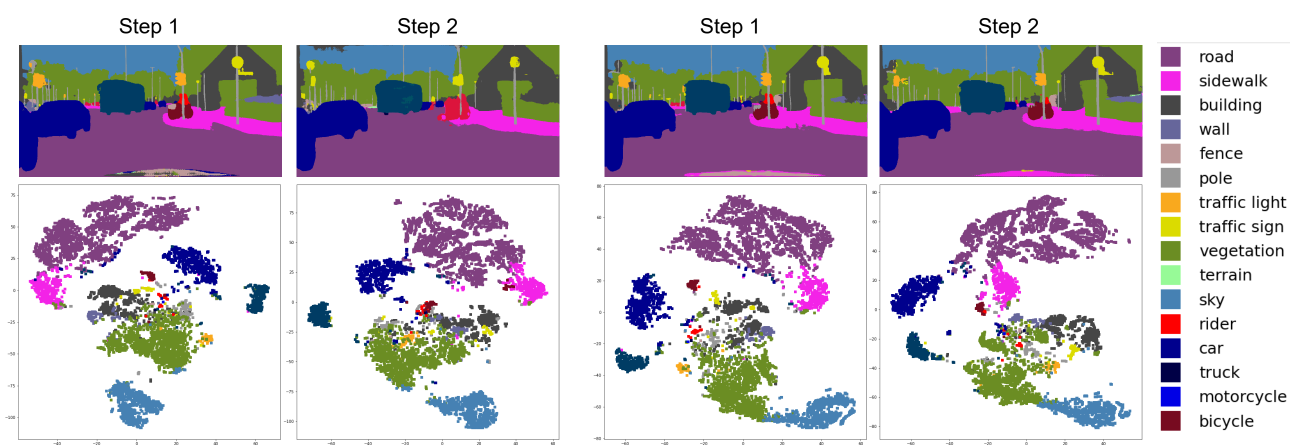
Figure A6: Confusion matrices plotted for CS validation set (a) After incrementally learning from $CS \rightarrow BDD$, (b) After incrementally learning from $CS \rightarrow IDD$.



(a) Fine-tuning (b) Ours



(a) Fine-tuning (b) Ours



(a) Fine-tuning (b) Ours

Figure A7: T-SNE plots, similar to Figure 5 of main paper

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[2] Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. Reparameterizing Convolutions for Incremental Multi-Task Learning without Task Interference. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–707. Springer, 2020.

[3] S-A Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[4] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8119–8127, 2018.

[5] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2017.

[6] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2636–2645, 2020.