HAD: Heterogeneity-Aware Distillation for Lifelong Heterogeneous Learning

Supplementary Material

A. Comparison with Existing Lifelong Learning Scenarios

In this section, we analyze the similarities and differences between the proposed lifelong heterogeneous learning for dense prediction (LHL4DP) scenario and the traditional lifelong learning scenario.

Similarities. The proposed LHL4DP scenario shares three key similarities with the traditional lifelong learning scenario: objectives, settings, and challenges. First, both LHL4DP and lifelong learning aim to achieve performance on sequential tasks comparable to that of joint training across multiple tasks [44]. Second, in both settings, models can be trained for multiple epochs on all data for a given task, while data from previous and future tasks remains inaccessible. Finally, both LHL4DP and lifelong learning face the issue of catastrophic forgetting, where training on the current task leads to the loss of knowledge from previous tasks.

Differences. Traditional lifelong learning can be classified into three subcategories: class incremental learning (CIL), task incremental learning (TIL), and domain incremental learning (DIL). Compared with other subcategories, the domain of the training data in DIL varies across tasks, while the number of classes remains consistent across different tasks. In both CIL and TIL, the domain of the training data remains consistent; however, as the number of tasks increases, so does the total number of classification categories. The key difference between TIL and CIL is that TIL requires a task ID during inference. However, CIL, TIL, and DIL remain restricted to a single task type, *e.g.*, only classification tasks, and do not support scenarios involving sequentially arriving heterogeneous tasks. In contrast, the proposed LHL4DP assumes tasks share an input distribution but differ in output types (*e.g.*, class labels and continuous values), which introduces unique challenges for LHL4DP. Illustration of the comparison is provided in Fig. 6.

B. Challenges of LHL4DP

The unique challenges posed by the proposed LHL4DP scenario are listed as follows.

Heterogeneous tasks. The technical challenges inherent in LHL4DP are beyond those typically encountered in conventional lifelong learning scenarios. Different from traditional settings that focus on a single type of tasks (*e.g.*, classification or segmentation), LHL4DP requires learning different types of tasks at different training phases, where each task often involves distinct objective functions and heterogeneous outputs. This results in a more complex and challenging training process.

Heterogeneous knowledge. Different tasks require distinct and heterogeneous knowledge representations. For example, the depth estimation task requires a comprehensive understanding of 3D scenes, while the semantic segmentation task primarily relies on high-level structured semantic knowledge [9, 24, 58]. This divergence presents a challenge for mitigating catastrophic forgetting during the learning of new tasks, highlighting the necessity of strategies that facilitate effective knowledge transfer across heterogeneous tasks.

Fine-grained information. DP tasks involve producing pixel-level outputs that rely on rich fine-grained information, thereby posing additional challenges [62]. This complexity makes retaining previously learned knowledge particularly difficult, requiring strategies capable of preserving fine-grained representations and producing globally coherent outputs across sequential DP tasks.

C. Experiment Details

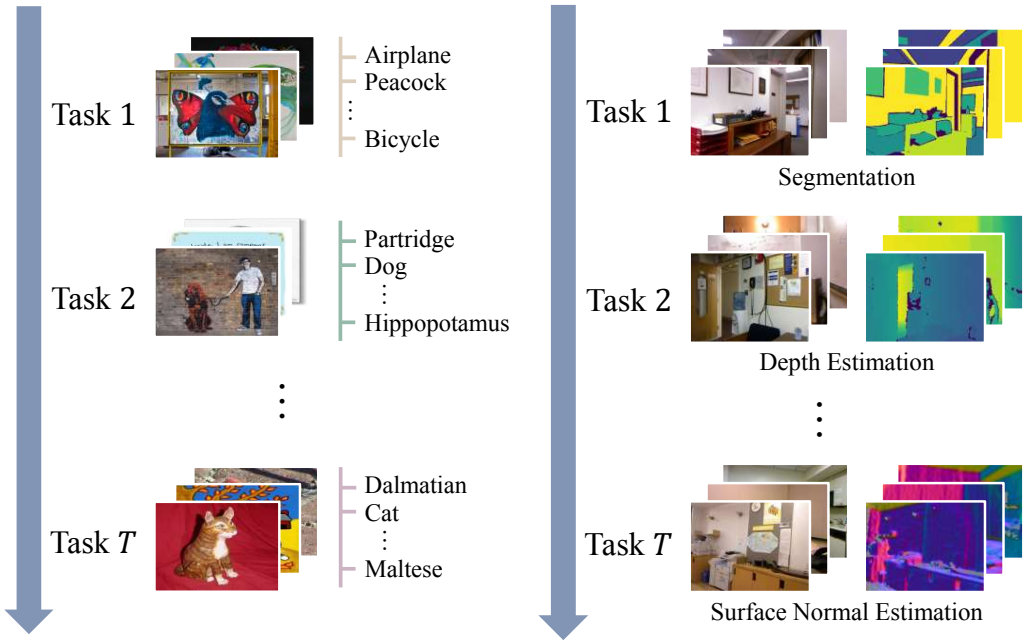
C.1. Details of datasets

To evaluate the performance of the proposed HAD method, we conduct experiments on three datasets with different task numbers as different scenarios: *NYUv2* dataset for 3 tasks, *CityScapes* dataset for 2 tasks, and *Taskonomy* dataset for 10 tasks.

***NYUv2* dataset.** This dataset contains 795 training images and 654 testing images in a variety of indoor scenes with ground truth for three tasks (*i.e.*, 13-class semantic segmentation, depth estimation, and surface normal prediction). We use the mean Intersection over Union (mIoU) and Pixel Accuracy (Pix Arr) to evaluate the semantic segmentation task, and use the Absolute Error (Abs Err) and the Real Error (Rel Err) to evaluate the depth prediction task. For the surface normal estimation task, it is evaluated with the mean and the median of angular error measured in degrees, and the percentage of pixels whose angular error is within 11.25, 22.5, and 30 degrees.

***CityScapes* dataset.** This dataset comprises 2,975 images for training and an additional 500 images for testing, where we conduct experiments on two tasks (*i.e.*, 7-class semantic segmentation and depth estimation). We use the mean Intersection over Union (mIoU) and Pixel Accuracy (Pix Arr) to evaluate the semantic segmentation task, and use the Absolute Error (Abs Err) and the Real Error (Rel Err) to evaluate the depth prediction task.

***Taskonomy* dataset.** We split the 1,390 images from three



(a) Lifelong Learning. (b) LHL4DP.

Figure 6. Comparison between Lifelong Learning and Lifelong Heterogeneous Learning for Dense Prediction (LHL4DP). (a) CIL incrementally recognizes all encountered classes, while (b) LHL4DP progressively addresses all encountered heterogeneous dense prediction tasks.

different views in this dataset into training data for the 10 tasks, reserving one unseen view for testing. The evaluation metric used for performance assessment is the test loss.

C.2. Heterogeneous Tasks

In the LHL4DP scenario, a new dense prediction task is introduced at each training phase. To accommodate this sequential heterogeneous setting, we employ a shared encoder across all tasks together with task-specific decoders. When a new task arrives, a new decoder is instantiated for it, while the decoders learned for previous tasks are retained. During each training phase, the current task is optimized with its corresponding task-specific loss $\mathcal{L}_{\text{new}}$, whose form depends on the task type. Tab. 5 summarizes the heterogeneous tasks in the *NYUv2*, *CityScapes*, and *Taskonomy* datasets, together with their task types and corresponding forms of $\mathcal{L}_{\text{new}}$.

C.3. Baselines

We compare the proposed HAD method against vanilla training, as well as three categories of traditional IL methods: regularization-based methods including EWC [25], LWF [28], and SGP [37], which constrain the changes in important parameters, representations, and gradients; replay-based methods such as iCaRL [34] and DER [4], which store historical data in a fixed-size memory and replay them during the learning of new tasks; and the parameter isolation method SPG [26], which combines orthogonal gradient projections with scaled gradient steps in the important gradient spaces for past tasks. For all replay-based methods, the exemplar size is fixed at 50.

For the hyperparameters of each method, we perform a grid search to select the best-performing configuration. Specifically, we adjust the loss balance weight for LWF, the penalty term for EWC, the distillation loss weights for iCaRL and DER, and the non-negative scale coefficient for SGP. The hyperparameters used for each method across different datasets are summarized in Table 6.

C.4. Implementation details

The task sequences are randomly selected. For the *NYUv2* dataset, the sequence is: Semantic segmentation → Depth estimation → Surface normal prediction, as shown in Tab. 1. For the *Taskonomy* dataset, the sequence is: Semantic segmentation (Seg.) → Depth estimation (Dep.) → Surface normal estimation (Normal) → Edge-3D detection (E.-3D) → Reshading (Res.) → Keypoint-2D detection (K.-2D) → Edge-2D detection (E.-2D) → Euclidean distance (E.D.) → Curvatures (Curv.) → Keypoint-3D detection (K.-3D), as shown in Tab. 2.

For all methods, we adopt the following common settings to ensure a fair comparison. The batch size is set to 64 for the *CityScapes* dataset, 16 for the *Taskonomy* dataset, and 48 for *NYUv2* dataset. We use the Adam optimizer with

Table 5. Task-specific forms of $\mathcal{L}_{\text{new}}$ for heterogeneous dense prediction tasks.

Dataset	Task Name	Task Type	Output Channel	Loss Function
NYUv2	Semantic Segmentation	Classification	13	Cross-Entropy Loss
	Depth Estimation	Regression	1	L_1 Loss
	Surface Normal Prediction	Regression	3	Cosine Distance Loss
CityScapes	Semantic Segmentation	Classification	7	Cross-Entropy Loss
	Depth Estimation	Regression	1	L_1 Loss
Taskonomy	Semantic Segmentation	Classification	8	Cross-Entropy Loss
	Depth Estimation (z-buffer)	Regression	1	L_1 Loss
	Surface Normal Estimation	Regression	3	Cosine Distance Loss
	Edge Occlusion Prediction	Regression	1	L_1 Loss
	Reshading	Regression	1	L_1 Loss
	Keypoint-2D Prediction	Regression	1	L_1 Loss
	Edge Texture Prediction	Regression	1	L_1 Loss
	Euclidean Depth Estimation	Regression	1	L_1 Loss
	Principal Curvature Estimation	Regression	2	L_1 Loss
	Keypoint-3D Prediction	Regression	1	L_1 Loss

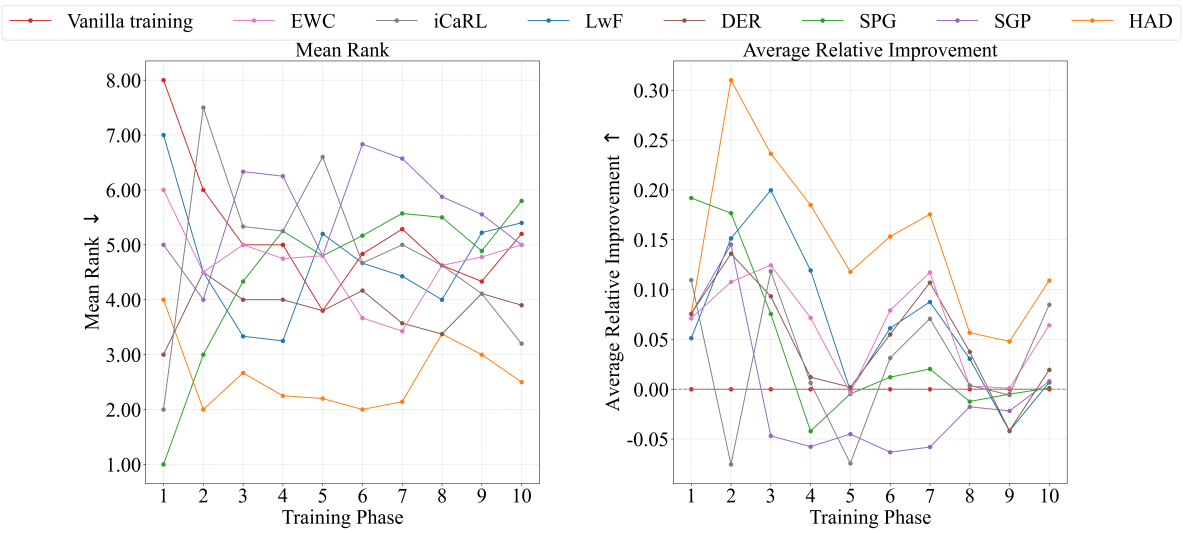


Figure 7. Results of different methods during each training phase on the *Taskonomy* dataset.

Table 6. Hyperparameter of different methods.

Method	*NYUv2*		*CityScapes*		*Taskonomy*
	Resnet-18	*Resnet-50*	*Resnet-18*	*Resnet-50*	*Resnet-18*
EWC	10^9	10^{10}	10^6	10^3	10^9
iCaRL	0.01	0.1	0.01	0.1	1
LWF	5	5	0.01	0.1	0.1
DER	0.01	0.1	1	1	0.1
SGP	0.1	10	10	1000	100

an initial learning rate of 10^{-4}, and adopt a linear learning rate scheduler with a warmup phase, where the warmup rate is set to 0.5. Weight decay is fixed at 10^{-5}.

In the proposed HAD method, we perform grid searches for the hyperparameters α, k, and τ. Specifically, we set hyperparameters as follows: $\alpha = 3, k = 0.5, \tau = 0.9$ for *NYUv2* dataset on *Resnet-18*, $\alpha = 20, k = 0.6, \tau = 0.6$ for *NYUv2* dataset on *Resnet-50*, $\alpha = 100, k = 0.6, \tau = 0.6$ for *CityScapes* dataset on *Resnet-18*, $\alpha = 1, k = 0.8, \tau = 0.5$ for *CityScapes* dataset on *Resnet-50*, and $\alpha = 1, k = 0.5, \tau = 0.9$ for *Taskonomy* dataset. We use the task-specific loss function as the per-pixel self-distillation loss function $\mathcal{L}_{\text{dis},j}$ of each task $\mathcal{T}_j$, *i.e.*, $\mathcal{L}_{\text{dis},j} = \mathcal{L}_j$.

All methods are implemented using the Pytorch framework, and all models are trained on RTX V100 GPUs.

D. Additional Results

The results for the shuffled task sequence on the *NYUv2* dataset are presented in Table 7. Sequence 2 consists of the following order: Surface Normal Prediction → Depth Estimation → Semantic Segmentation, while Sequence 3 follows the order: Depth Estimation → Semantic Segmen-

Table 7. Performance on the *NYUv2* dataset using *Resnet-18* with a shuffled task sequence after the last training phase. The best results for each task are shown in **bold**. ↑(↓) means that the higher (lower) the value, the better the performance.

	Method	Segmentation		Depth		Surface Normal					$\Delta_b^T \uparrow$	MR↓
		mIoU↑	Pix Acc↑	Abs Err ↓	Rel Err↓	Angle Distance		Within t°				
						Mean ↓	Median ↓	11.25 ↑	22.5 ↑	30 ↑		
Sequence 2	Vanilla training	33.77	60.36	1.0261	0.3592	40.76	34.74	10.16	30.98	43.17	+0.00%	6.00
	Joint training	41.84	66.14	0.5793	0.2201	31.53	25.78	22.38	44.54	56.36	+3.09%	1.00
	EWC	29.78	56.75	0.9217	0.3218	39.15	33.31	10.48	31.98	44.95	−0.77%	5.33
	iCaRL	23.87	53.78	1.5976	0.5474	35.87	33.19	11.60	31.55	44.63	−27.11%	7.67
	LwF	31.49	58.96	0.8586	0.3044	37.66	32.54	11.76	33.50	46.17	+0.77%	3.67
	DER	24.41	54.38	1.5884	0.5428	35.78	33.00	11.71	31.84	44.96	−26.55%	6.67
	SPG	34.40	60.79	1.0025	0.3454	40.01	34.18	11.10	31.86	43.96	+0.29%	4.67
	SGP	34.34	61.16	1.0859	0.3803	41.30	35.83	9.62	29.30	41.47	−0.17%	6.00
	HAD	28.86	56.83	0.7024	0.2553	35.05	30.36	12.71	36.04	49.41	**+1.18%**	**3.33**
Sequence 3	Vanilla training	22.05	51.48	1.0131	0.3709	33.70	28.22	16.43	39.92	52.86	+0.00%	5.00
	Joint training	41.84	66.14	0.5793	0.2201	31.53	25.78	22.38	44.54	56.36	+28.98%	1.00
	EWC	19.60	49.40	0.9408	0.3476	38.45	34.20	11.52	31.50	43.66	+6.62%	7.33
	iCaRL	15.35	49.91	1.2300	0.4093	38.33	33.66	10.53	31.16	44.15	−3.71%	8.33
	LwF	23.46	55.26	0.8928	0.3239	37.21	32.05	16.41	35.68	47.04	+9.43%	3.67
	DER	15.14	49.87	1.2280	0.4076	38.17	33.48	10.63	31.41	44.41	−3.96%	8.00
	SPG	20.98	51.08	0.9314	0.3364	34.38	28.86	15.16	38.69	51.79	+3.10%	5.00
	SGP	22.18	52.15	0.8878	0.3245	32.63	27.47	19.29	41.63	53.96	+2.49%	**3.00**
	HAD	25.67	54.85	0.8358	0.2992	36.96	31.81	15.81	35.83	47.39	**+13.08%**	**3.00**

Table 8. Performance on two tasks after the last training phase (i.e., 7-class semantic segmentation and depth estimation) of the *CityScapes* dataset using *ResNet-18* under two different sequences.

Method	*Segmentation → Depth*						*Depth → Segmentation*					
	Segmentation		Depth		$\Delta_b^T \uparrow$	MR↓	Segmentation		Depth		$\Delta_b^T \uparrow$	MR↓
	mIoU↑	Pix Acc↑	Abs Err ↓	Rel Err↓			mIoU↑	Pix Acc↑	Abs Err ↓	Rel Err↓		
Vanilla training	58.40	86.84	0.0203	50.0861	+0.00%	6.50	68.44	91.45	0.0456	77.8347	+0.00%	5.50
Joint training	71.38	92.15	0.0164	43.7236	+15.06%	1.00	71.38	92.15	0.0164	43.7236	+28.23%	1.00
EWC	66.14	90.01	0.0203	56.8526	+0.85%	6.50	65.56	90.06	0.0217	51.9560	+19.98%	5.50
iCaRL	67.10	91.09	0.0204	50.2167	+4.76%	5.50	57.24	87.82	0.0221	58.8481	+13.90%	7.00
LwF	62.67	87.98	0.0202	47.6400	+3.50%	4.50	67.52	89.54	0.0192	47.2005	+24.91%	5.00
DER	67.15	91.11	0.0206	51.3117	+3.99%	5.50	69.20	91.90	0.0256	54.3059	+18.92%	**3.50**
SPG	68.07	91.31	0.0484	94.8094	−51.50%	5.50	69.68	91.75	0.0418	104.0395	−5.80%	5.00
SGP	53.00	82.38	0.0202	49.2638	−3.06%	6.00	68.07	91.31	0.0484	94.8094	−7.16%	7.50
HAD	68.28	90.76	0.0192	51.7254	**+5.89%**	**3.50**	69.12	91.12	0.0186	45.5291	**+26.70%**	**3.50**

tation → Surface Normal Prediction. As observed, the proposed HAD method consistently outperforms the baseline approaches, further validating its effectiveness, regardless of the task sequence.

The results of the *CityScapes* dataset using *ResNet-18* with different task sequences are provided in Tab. 8. Tab. 9 presents the results of the proposed HAD method on the same dataset using *ResNet-50*, with semantic segmentation as the first task and depth estimation as the second. As can be seen, the proposed HAD method outperforms baseline methods in both mitigating the performance degradation of the previous task and improving overall performance. Note that although the *Cityscapes* dataset contains only 2 tasks, the LHL4DP scenario is fundamentally different from transfer learning (TL), as LHL4DP treats all tasks equally by preserving the performance of previous tasks. In contrast, TL primarily focuses on optimizing the performance of the target task.

Additionally, in Fig. 7, we present the results of average relative improvement (Δ_v^m) and mean rank (**MR**), detailed in Sec. 5.1, across various methods in each training phase on the *Taskonomy* dataset. As can be seen, the proposed method HAD consistently outperformed the baseline methods.

Table 9. Performance on two tasks after the last training phase (i.e., 7-class semantic segmentation and depth estimation) of the *CityScapes* dataset using *Resnet-50*. The best results for each task are shown in **bold**. ↑(↓) means that the higher (lower) the value, the better the performance.

Method	Segmentation		Depth		$\Delta_b^T \uparrow$	MR↓
	mIoU↑	Pix Acc↑	Abs Err ↓	Rel Err↓		
Vanilla training	64.93	88.93	0.0168	40.0604	+0.00%	5.50
Joint training	76.49	93.91	0.0155	45.7162	+4.26%	3.00
EWC	69.44	90.60	0.0154	41.4725	+3.41%	**3.50**
iCaRL	72.80	92.60	0.0166	47.3969	−0.22%	7.00
LwF	76.07	93.66	0.0175	43.5094	+2.42%	5.00
DER	73.26	93.16	0.0159	46.5341	+1.70%	6.00
SPG	57.99	85.69	0.0155	44.7024	−4.55%	6.00
SGP	65.53	88.44	0.0157	43.8087	−0.61%	4.50
HAD	76.52	93.81	0.0165	44.2749	**+3.65%**	**3.50**

Table 10. Performance on 3 tasks (*i.e.*, 13-class semantic segmentation, depth estimation, and surface normal prediction) after the last training phase of the *NYUv2* dataset compared with feature distillation methods. The best results are shown in **bold**. ↑(↓) means that the higher (lower) the value, the better the performance.

Method	Segmentation		Depth		Surface Normal					$\Delta_v^m \uparrow$	MR↓
					Angle Distance		Within t°				
	mIoU↑	Pix Acc↑	Abs Err ↓	Rel Err↓	Mean ↓	Median ↓	11.25 ↑	22.5 ↑	30 ↑		
Vanilla training	17.49	46.81	0.9609	0.3328	32.45	26.92	20.72	42.56	54.73	+0.00%	5.67
Joint training	41.84	66.14	0.5793	0.2201	31.53	25.78	22.38	44.54	56.36	+40.83%	1.00
Local POD w. fro	32.25	58.11	0.8651	0.3063	35.43	30.40	16.15	37.23	49.36	+25.51%	3.33
Local POD w. L_1	32.56	58.70	0.9240	0.3289	36.70	32.22	14.72	34.67	46.62	+25.77%	5.00
Local POD w. L_2	32.21	58.06	0.8658	0.3061	35.56	30.49	15.78	37.08	49.22	+25.66%	5.00
Global POD w. fro	32.31	58.14	0.8698	0.3077	35.47	30.41	16.19	37.24	49.34	+25.42%	4.33
Global POD w. L_1	32.30	58.12	0.8668	0.3067	35.45	30.40	16.07	37.24	49.36	+25.54%	3.33
HAD	35.12	59.63	0.7410	0.2641	35.32	30.55	17.23	37.26	49.12	**+32.74%**	**2.00**

E. Additional Ablation

To further demonstrate the effectiveness of HAD against feature distillation baselines, we compare it with the Local POD and Global POD feature distillation strategies adopted in PLOP [15] on the *NYUv2* dataset. As shown in Tab. 10, all POD-based variants consistently outperform vanilla training, indicating that feature distillation is beneficial in the LHL4DP setting. Nevertheless, they remain inferior to HAD, which demonstrates the superiority of the proposed method in heterogeneous lifelong dense prediction tasks.

F. Hyperparameter Sensitivity Analysis

In this section, we investigate the impact of hyperparameters on the performance of the proposed HAD method, evaluated on the $NYUv2$ dataset. Specifically, we explore the effects of the distillation weight (α), the region partition threshold (k), and the Sobel threshold (τ). The results for varying distillation weights (α) are provided in Tab. 12, while Tab. 11 presents the performance variations for different values of τ and k. The results presented in both tables illustrate the performance differences relative to the results of HAD reported in Tab. 1.

Table 11. Δ_b^T of different hyperparameter k, τ.

k \ τ	0.5	0.6	0.7	0.8	0.9
0.5	-4.87	-1.07	-2.13	-0.92	0.00
0.6	-2.61	-2.66	-0.12	-1.76	-0.47
0.7	-3.58	-3.48	-1.21	-1.77	-0.81
0.8	-4.07	-2.41	-2.11	-2.11	-1.48
0.9	-5.25	-3.87	-2.44	-2.28	-2.31

Table 12. Δ_b^T of different hyperparameter α.

α	1	2	3	4	5	6
$\Delta_b^T \uparrow$	-5.79	-2.71	0.00	-0.60	-1.10	-0.83

The performance of the proposed HAD method remains

consistently stable across a reasonable range of hyperparameter values. For example, the proposed method can achieve satisfactory results with k in 0.5-0.6 and τ in 0.7-0.9.