

Technical Appendices and Supplementary Material

A Detailed Experimental Settings

Table A1: Dataset configurations for pre-training, incremental, and zero-shot test datasets used in the four different learning sequences.

Type of Learning Sequence	Pre-training Dataset	Incremental Dataset	Zero-shot test Dataset
(S1) Scenario 1	COCO	Cityscapes	ADE20K
(S2) Scenario 2	COCO	ADE20K	Cityscapes
(S3) Scenario 3	COCO	Cityscapes, ADE20K	LVIS, BDD100K, Mapillary Vistas, PC-59, PC-459, PAS-20, PAS-21, A-847
(S4) Scenario 4	COCO	Cityscapes, ADE20K, BDD100K, Mapillary Vistas	LVIS, PC-59, PC-459, PAS-20, PAS-21, A-847

This study assumes scenarios where trainable datasets are provided sequentially and evaluates the performance of OVS models that are incrementally trained on these datasets. Specifically, we first train an OVS model from scratch using the pre-training dataset. In the case of fc-clip, only the decoder is trained from scratch, whereas X-Decoder trains both the encoder and decoder. After pre-training, we incrementally train the model on the training sets of each incremental dataset in sequence. As shown in Table A1, we define four experimental scenarios (S1, S2, S3, S4) based on different learning sequences of the datasets. Finally, the model is evaluated using the evaluation sets of both the pre-training and incremental datasets, as well as zero-shot test datasets.

A.1 Datasets

Pre-training Datasets. In our experiments, we follow the pre-training dataset configurations proposed in each OVS model. Specifically, fc-clip uses only the COCO [21] dataset for pre-training, while X-Decoder is trained on COCO for segmentation and additionally uses four image-text pair datasets. Since COCO is the only segmentation dataset used in the training of both models, we evaluate the segmentation performance on the pre-training dataset using the COCO evaluation set.

Incremental Datasets. We use Cityscapes [5] and ADE20K [36] as incremental datasets. Cityscapes is specialized for urban driving scenes, whereas ADE20K covers a wide range of fine-grained things and stuff classes from both indoor and outdoor environments. In scenarios S1 and S2, where only one of these datasets is used for incremental learning, the other is used for zero-shot testing. For example, when Cityscapes is designated as the incremental dataset, ADE20K serves as the zero-shot test dataset.

Zero-shot Test Datasets. To evaluate the generalization performance of the OVS model on unseen classes not included during training, we use a total of eight datasets: LVIS [7], BDD100K [34], Mapillary Vistas [26], PC-59, PC-459, PAS-20, PAS-21, and A-847. Here, **PC** refers to Pascal Context [24], **PAS** to Pascal VOC [6], and **A** to ADE20K. The number following each name indicates the number of classes included in the corresponding dataset.

A.2 Implementation Details

We apply the proposed method to two OVS models: fc-clip with ConvNeXt-L [22] and X-Decoder with Focal-L [33]. When fine-tuning on incremental datasets, we follow the original fine-tuning protocols of fc-clip and X-Decoder, in which the encoder is frozen and only the decoder is updated. The temperature parameter T used in the softmax operation for estimating interpolation factors is set to 0.01. When computing probabilities from the MVN distributions, we use the log-likelihood. All experiments are conducted using two NVIDIA A5000 GPUs.

A.3 Evaluation Metrics

The two baselines used in this study, fc-clip and X-Decoder, are universal segmentation models capable of performing panoptic, instance, and semantic segmentation. Accordingly, we evaluate model performance across all three tasks using the following metrics: (1) Panoptic Segmentation

is evaluated with Panoptic Quality (PQ), (2) Instance Segmentation with mean Average Precision (mAP), and (3) Semantic Segmentation with mean Intersection over Union (mIoU). All PQ, mIoU, and mAP values are reported in Table A14, A15, and A16, where these three metrics show consistent performance trends. In addition, some zero-shot test datasets are limited to specific segmentation types. For instance, LVIS supports only instance segmentation, while PC-459 provides only semantic segmentation annotations. In such cases, we evaluate performance using only the supported task for each dataset.

B Compared Methods

B.1 Overview of Adapted Continual Learning Methods for OVS

Existing continual learning (CL) methods are known to be unsuitable for open-vocabulary tasks, as they are based on the assumption that only classes included in the training dataset can be recognized. In this paper, we analyze how this limitation affects performance by comparing the proposed method with existing CL approaches.

To this end, we adapt four representative CL methods to be compatible with the OVS model. Following prior works [2, 25, 27, 31], we categorize existing CL methods into three groups: replay-based, regularization-based (parameter, function), and parameter-isolation-based. We select representative methods from each category and apply it to the OVS model. Section B.2 to B.4 describes the overview of each method and the modifications made to align them with the OVS setting.

B.2 Replay-based Method: ER

Experience Replay (ER) stores a fixed number of samples per class from previous training datasets and uses them when training on new datasets. This technique serves as the conceptual foundation for various memory-based continual learning methods [14, 23].

To apply ER to the OVS model, we select ten samples per class from the previous training dataset. Since OVS uses a multi-label structure where each image can contain multiple classes, we prevent duplicate selection by sequentially sampling ten images per class without redundancy.

ER assumes access to the pre-training dataset during the training of new datasets. Therefore, there is a limitation in conducting a fair comparison with the proposed method or other CL methods that do not require access to pre-training data. Nevertheless, we include ER in our comparison to provide a broad performance analysis across different CL strategies.

B.3 Regularization-based Methods: LwF & EWC

Regularization-based methods include two subtypes: (1) function regularization and (2) parameter regularization. For function regularization, we apply Learning without Forgetting (LwF) [20] to the OVS model. LwF uses a knowledge distillation loss based on the prediction distance between the previously trained model and the newly trained model. Since LwF constructs its loss using the probability scores generated by the previously trained model, it requires the preservation of a classification head. However, OVS models do not include classification heads. To address this, we instead compute the similarity between the text embedding of the previously trained model and the class embeddings to obtain probability scores used for distillation.

Second, we apply Elastic Weight Consolidation (EWC) [17], a parameter regularization method, to the OVS model. EWC estimates the Fisher Information Matrix using the new training dataset, which measures the importance of each parameter. It then penalizes changes to important parameters, thereby preserving previously learned knowledge. Since this method does not alter the architecture of the OVS model, it can be applied without structural modifications. However, the effectiveness of this approach depends on clear separation between datasets. If there are overlapping or similar classes or domains between the datasets, integrating knowledge from both can improve performance. EWC, however, restricts updates to parameters deemed important for the first dataset, which hinders the model’s ability to incorporate new knowledge from the second dataset. As a result, even when the datasets share useful information, the model cannot integrate it effectively, making EWC less suitable for scenarios involving sequential dataset training.

B.4 Parameter-isolation-based Method: ECLIPSE

We apply ECLIPSE [16], a parameter-isolation-based method, to the OVS model. ECLIPSE is designed for class-incremental learning in closed-set segmentation settings. Specifically, it adds learnable prompts to the object queries and positional embeddings when learning new classes, and updates the classification head accordingly. Since OVS models do not use a classification head, we exclude the classification head component and apply only the prompt tuning elements.

ECLIPSE also utilizes logit manipulation based on class-wise probability scores obtained from the classification head. This component helps determine whether the predicted mask from an object query corresponds to a valid class or to a “no object” case, preventing semantic drift for unseen classes in closed-set settings. However, the OVS model must recognize unseen classes and does not include a classification head, which makes direct application of the logit manipulation method infeasible. Therefore, we also exclude the logit manipulation component of ECLIPSE in our implementation.

Lastly, because each incremental dataset contains a large number of classes, we assume that a sufficient number of learnable parameters is necessary. Accordingly, we configure the model to learn 250 additional prompts per dataset.

B.5 Alternative Approaches to MVN Distribution

K-Means Clustering is an unsupervised learning algorithm that partitions a given dataset into K clusters. Each cluster is associated with a centroid, and the algorithm iteratively assigns each data point to the cluster whose centroid is closest, then updates the centroid as the mean of all data points assigned to the cluster. Clustering is based on the Euclidean distance, and the algorithm aims to minimize the sum of squared distances between data points and their corresponding centroids. K-Means is computationally efficient and easy to implement. However, it struggles with robustness under noisy conditions and cannot effectively model non-spherical or overlapping distributions.

Kernel Density Estimation (KDE) is a non-parametric method for estimating the probability distribution of a given dataset. KDE constructs the probability density function by placing a kernel function on each data point and summing their contributions. In addition, a bandwidth parameter in KDE controls the smoothness of the resulting density: small bandwidths lead to overfitting, while large bandwidths oversmooth the distribution. For that reason, we set the bandwidth to 0.5. Due to its flexibility, KDE can approximate arbitrary distributions without requiring any parametric assumptions. However, unlike the MVN distribution, KDE does not offer a compact parametric form. As a result, it suffers from unstable likelihood estimation, making it less suitable for tasks that rely on density-based inference, such as interpolation factor estimation.

C Details of Task Vectors

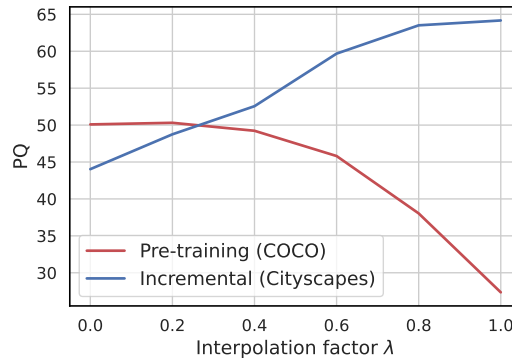


Figure C1: Performance on the evaluation set of Cityscapes and COCO depending on the interpolation factor λ , using fc-clip.

The task vector [13] is constructed by subtracting the weights of a pre-trained model from those of a model fine-tuned on a specific task. These task vectors can be modified or combined through

arithmetic operations such as addition, and the behavior of the resulting model is adjusted accordingly. For example, by adding two task vectors obtained from fine-tuning on different tasks and then adding the result to the pre-trained model’s weights, one can generate model weights that can be utilized on both tasks. In addition, an interpolation factor λ can be multiplied by the task vector. This λ determines whether the model uses the weights trained on the pre-training dataset or those trained on the fine-tuning dataset.

To verify whether the combination of interpolation factors and task vectors is also effective in OVS models, we construct a task vector by subtracting the weights of a pre-trained OVS model from those of a fine-tuned OVS model and evaluate the performance changes when various λ values are multiplied to this vector. As shown in Figure C1, when $\lambda = 0$, the decoder uses the pre-trained weights $\theta_{\text{dec,pr}}$, which results in strong performance on the pre-training dataset. In contrast, when λ is close to 1, the decoder uses weights close to the fine-tuned weights $\theta_{\text{dec,ft}}$, leading to high performance on the fine-tuning dataset. When λ takes a value between 0 and 1, the decoder interpolates between the two weights and achieves balanced performance across both datasets.

D Computational Resources

D.1 Training Resources

Table A2: Comparison of training time and GPU memory usage across different methods. The values are reported relative to standard fine-tuning. For reference, fine-tuning required 5110 seconds and 22.7 GB in Scenario 1, and 5701 seconds and 20.8 GB in Scenario 2.

Method	Scenario 1 (Cityscapes)		Scenario 2 (ADE20K)	
	Training Time	GPU Memory	Training Time	GPU Memory
Fine-tuning	1.00	1.00	1.00	1.00
Retraining	1.25	0.86	1.40	0.95
ER	1.25	0.86	1.40	0.95
LwF	1.28	1.07	1.21	1.13
EWC	1.07	1.00	1.01	1.03
ECLIPSE	0.75	0.57	0.74	0.52
ConOVS (ours)	1.00	1.00	1.00	1.00

To provide a fair comparison, we measured the training time and GPU memory usage of each method and normalized all results to the computational cost of fine-tuning. As shown in Table A2, training time does not vary substantially across methods, which is expected since the number of training iterations was fixed for all experiments. An exception is ECLIPSE, which demonstrates lower training cost due to its use of Visual Prompt Tuning, updating only a small subset of model parameters. While this design improves training efficiency, our results show that ConOVS achieves superior segmentation performance compared to ECLIPSE. We attribute this performance gap to the inherent limitations of methods that update only a fraction of the model parameters.

D.2 Inference Resources

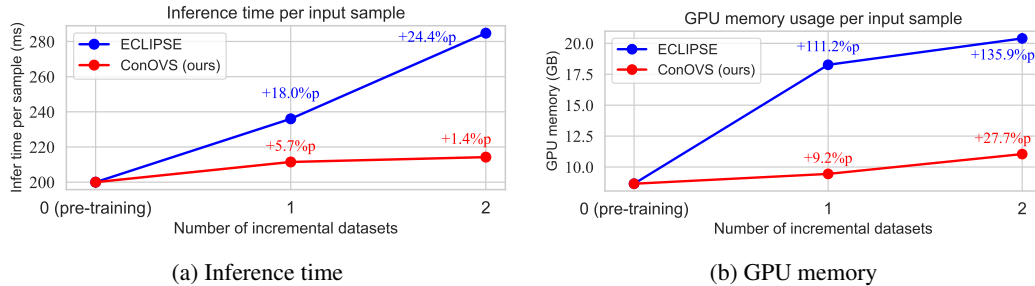


Figure D2: Comparison of per-sample inference resources between ConOVS (ours) and ECLIPSE.

Our method is based on a Mixture-of-Experts (MoE) framework that dynamically combines multiple expert models according to the input sample. In MoE-based continual learning [18], a new expert

is added each time an incremental dataset is introduced. This raises concerns that the number of parameters involved in inference may increase with the number of incremental datasets, potentially leading to higher inference time and GPU memory usage [9]. To examine this issue, we measure how resource usage scales with the number of incremental datasets and compare our method against ECLIPSE [16], a parameter-isolation-based method that adds parameters during incremental learning.

As shown in Figure D2, our method consistently requires less inference time and GPU memory than ECLIPSE. In addition, as the number of incremental datasets increases, the growth in inference time is significantly smaller for our method. This difference arises from how the additional parameters are utilized during inference. In ECLIPSE, all parameters added during training are used at inference, so the number of parameters used increases in proportion to the number of incremental datasets. In contrast, our method interpolates expert weights based on the input sample to generate a single decoder weight, so the number of parameters used during inference remains constant regardless of the number of trained datasets.

Table A3: Inference time per sample with varying numbers of incremental datasets. The unit for all numbers in the table is milliseconds (ms).

Number of Incremental Datasets	Encoder	Interpolation Factor Estimator	Expert Interpolation	Decoder	Total Inference Time Per Sample	Change (%)
0	97.69	-	-	102.30	199.99	+0.00%
1	97.69	0.81	10.69	102.30	211.48	+5.75%
2	97.69	1.01	13.23	102.30	214.23	+7.12%

In practice, our method only incurs additional resource usage when estimating the interpolation factor or interpolating the expert weights; otherwise, the resource usage of the model itself remains unchanged. As shown in Table A3, the inference time of the encoder and decoder parts does not increase even as the number of incremental datasets grows.

Beyond inference time, our method also achieves high efficiency in terms of storage. Unlike ensemble-based approaches [15, 32], which require storing the full model weights to combine multiple models, our method stores only the parameters of the decoder. As a result, it is sufficient to store only 6.11% of the total model size, which corresponds to approximately 80MB per dataset. This efficiency ensures high scalability even in scenarios where the number of models to be combined gradually increases.

E More Incremental Datasets

Table A4: Performance comparison across pre-training, incremental, and zero-shot datasets when sequentially training on five datasets. The best performance for each dataset is highlighted in bold.

Method	Pre-training	Incremental					Zero-shot					Average
	COCO	Cityscapes	A-150	BDD100K	Mapillary	LVIS	PC-59	PC-459	PAS-20	PAS-21	A-847	
fc-clip	50.1	44.0	23.5	19.0	26.0	20.5	53.0	16.9	93.1	80.2	13.8	40.0
Fine-tuning	31.7	55.5	28.7	25.8	36.3	17.7	49.8	16.3	90.0	73.9	14.0	40.0
Retraining	49.0	62.3	36.3	28.2	34.5	21.7	53.6	17.4	94.0	80.7	15.6	44.8
ConOVS (ours)	50.1	62.9	38.5	29.1	35.0	21.8	53.6	17.3	93.2	80.4	16.0	45.3

To validate the effectiveness of our method on a larger number of incremental datasets, we designed Scenario 4 (S4) in which the model is sequentially trained on five datasets: COCO, Cityscapes, ADE20K, BDD100K, and Mapillary Vistas. Although we aimed to include additional datasets, we were constrained to those that provide panoptic segmentation annotations.

The results of this extended experiment are presented in Table A4. As shown, our method outperforms the baseline, fine-tuning, and retraining approaches. These findings demonstrate that our method remains effective as the number of incremental datasets increases, highlighting its scalability under complex domain shifts.

F Additional Ablation Studies of ConOVS

F.1 Ablation Study on Fine-tuned Components

Table A5: Ablation study on different fine-tuned components. The best performance for each dataset is highlighted in bold.

Fine-tuned Component	COCO (pre-training)	Cityscapes (incremental)	ADE20K (zero-shot)
fc-clip	50.1	44.0	23.5
Last Block	50.0	50.4	25.8
LayerNorm	49.9	44.0	25.8
LoRA	50.7	47.9	25.9
Only decoder	50.4	64.4	26.0

Given recent works that fine-tune the CLIP encoder [4, 19], a more detailed ablation study on our design choice to freeze the encoder and fine-tune only the decoder is necessary. To this end, we experimented with three encoder fine-tuning strategies: (1) fine-tuning only the last block of the encoder, (2) fine-tuning only the LayerNorm modules, and (3) replacing all MLP layers with LoRA modules and fine-tuning them. Note that, unlike prior attention-based methods designed for ViT-style CLIP encoders [4, 19], our strategies are tailored to the ConvNeXt-based encoders used in fc-clip.

As shown in Table A5, all three encoder fine-tuning strategies yield lower performance on the incremental dataset compared to our design choice of fine-tuning only the decoder. We attribute this to an architectural difference: whereas previous methods [4, 19] generate segmentation masks directly from the encoder and thus benefit from encoder fine-tuning, fc-clip generates masks in the decoder’s mask head. This suggests that decoder fine-tuning is more effective for improving segmentation performance in our setup.

F.2 Ablation Study of Softmax Operation

Table A6: Performance comparison with different normalization operations. The best results for each column are highlighted in bold.

Operation	COCO (pretraining)	Cityscapes (incremental)	ADE20K (zero-shot)	Average
minmax	50.3	50.5	26.1	42.3
sigmoid	50.3	50.3	26.0	42.2
softmax	50.4	64.4	26.0	46.9

We apply the softmax operation to the log-likelihood vector to normalize the proximity scores of each domain to the $[0, 1]$ range. This decision is based on a prior study [13], which found that when the interpolation factor exceeds 1, merging performance degrades. To normalize the log-likelihood values obtained from the MVN distribution, we considered three strategies, including min-max normalization, sigmoid, and softmax. As shown in Table A6, softmax achieved the best performance, which led us to adopt it.

F.3 Ablation Study of Element-wise Maximum Operation

Table A7: Comparison of operations for calculating the interpolation factor λ , with the best result in each column highlighted in bold.

Input	Operation	COCO (pre-training)	Cityscapes (incremental)	ADE20K (zero-shot)	Average
image only	-	51.5	43.4	25.8	40.2
text only	-	51.9	60.7	25.9	46.2
image+text	Average	50.3	64.2	25.9	46.8
image+text	Multiplication	50.1	63.8	25.8	46.6
image+text	Maximum	50.4	64.4	26.0	46.9

We use the element-wise maximum operation to combine information from both the image and text modalities. Initially, we considered whether to rely on only the image domain, only the text domain, or both. For combining both modalities, we evaluated three options: average, multiplication, and element-wise maximum.

As shown in Table A7, the results show that combining both modalities performs better than using a single modality, and among the combination methods, element-wise maximum achieved the best performance. Based on these findings, we selected element-wise maximum as our fusion strategy.

G Additional Analysis

G.1 Evaluation on Diverse and Challenging Domains

Table A8: Performance comparison (mIoU) on datasets with significant domain shifts.

Method	GTA5	DarkZurich	FoggyZurich
fc-clip	65.6	40.2	54.4
+ Fine-tuning	58.4	39.8	52.1
+ ConOVS (ours)	66.6	43.1	55.9

To demonstrate the robust generalization capability of the proposed method, we evaluate the model on three zero-shot test datasets that differ significantly from the domains of the training datasets. Specifically, we train the model using COCO as the pre-training dataset and ADE20K as the incremental dataset. For evaluation, we use GTA5 [29] (a synthetic driving simulation), DarkZurich [30] (nighttime driving scenes), and FoggyZurich [8] (driving scenes with fog), which represent domains that are substantially different from the training data.

As shown in Table A8, simply fine-tuning fc-clip on the incremental dataset leads to performance degradation across all three zero-shot test datasets. In contrast, the proposed method improves performance on all of them, demonstrating its effectiveness even under adverse conditions and in synthetic environments with large domain shifts.

G.2 Different Pre-training Dataset

Table A9: Performance comparison among fc-clip, fine-tuning, and ConOVS (ours), with ADE20K as the pre-training dataset. The incremental dataset is either (a) COCO or (b) Cityscapes. PQ is used.

Method	ADE20K (pre-training)	COCO (incremental)	Cityscapes (zero-shot)
fc-clip	48.1	42.3	40.9
Fine-tuning	-18.5	+10.4	+3.3
ConOVS (ours)	-1.3	+9.3	+5.2

(a) COCO

Method	ADE20K (pre-training)	Cityscapes (incremental)	COCO (zero-shot)
fc-clip	48.1	40.9	42.3
Fine-tuning	-18.5	+21.4	-11.5
ConOVS (ours)	+0.0	+19.5	+0.0

(b) Cityscapes

To verify the effectiveness of the proposed method on OVS models pre-trained on datasets other than COCO, we apply our method to an OVS model pre-trained on ADE20K and compare the performance. As shown in Table A9, even when the pre-training dataset changes, the proposed method significantly improves performance on the incremental dataset compared to the baseline fc-clip (e.g., +19.5 on Cityscapes). Moreover, compared to fine-tuning, the proposed method maintains the performance on the pre-training dataset (e.g., on ADE20K, Fine-tuning: -18.5, ConOVS: +0.0) while achieving strong performance on the incremental dataset. These results suggest that the proposed method can effectively expand the recognition capability of OVS models by learning newly collected datasets, regardless of the type of pre-training dataset.

G.3 Performance of Retraining Across Training Iterations

As shown in Section 6, the retraining method demonstrates relatively lower performance. This is because all methods were conducted under the same computational budget. Since retraining requires learning from a substantially larger amount of data compared to our method, it needs a longer training schedule to reach convergence.

To analyze this more precisely, we conducted additional experiments by training the retraining method for longer durations. As presented in Table A10, when trained with a sufficiently long schedule (100k iterations), the retraining method achieves better performance than our approach on the pretraining

Table A10: Performance comparison across training iterations of the retraining. PQ is used.

Method	Training Iterations	COCO (pre-training)	Cityscapes (incremental)	ADE20K (zero-shot)
Retraining	10k	50.7	61.9	25.2
Retraining	20k	50.7	62.5	25.4
Retraining	40k	51.0	63.3	25.3
Retraining	100k	51.0	64.4	25.5
ConOVS (ours)	10k	50.4	64.2	26.0

and incremental datasets. However, despite consuming significantly more computational resources, retraining does not provide a substantial improvement over ConOVS. One possible explanation for this limited performance is Task Interference [1, 35], which may occur when training on multiple domains simultaneously. In contrast, ConOVS avoids this issue by independently training domain-specific experts and dynamically combining them at inference time. Consequently, our method achieves performance comparable to retraining while requiring significantly less computational cost.

G.4 Effectiveness of ConOVS with Multi-Domain Incremental Datasets

Table A11: Comparison between the baseline and ConOVS when the incremental dataset consists of multiple domains.

Method	COCO (pre-training)	Cityscapes (incremental)	ADE20K (incremental)	Average on 4 zero-shot datasets
fc-clip	50.1	44.0	23.5	60.8
ConOVS (ours)	50.1	60.4	31.5	61.3

We conducted an additional experiment to assess whether our method remains effective when a single incremental dataset contains samples from multiple domains. For this purpose, we combined Cityscapes and ADE20K into a single incremental dataset. Note that the performance on the zero-shot setting was measured by averaging results across four datasets: PC-59, PC-459, PAS-20, and PAS-21.

As shown in Table A11, our method maintains the performance of the baseline fc-clip on the pre-training dataset, while significantly improving results on both the incremental and zero-shot datasets. These findings demonstrate that ConOVS remains robust and effective even when the incremental dataset spans diverse domains.

G.5 Effectiveness of ConOVS in Low-Resource Incremental Settings

Table A12: Performance comparison between fc-clip and ConOVS with a small incremental dataset (5% of Cityscapes). ConOVS maintains pre-training performance while improving incremental and zero-shot results, showing robustness in low-resource settings.

Method	COCO (pre-training)	Cityscapes (incremental)	ADE20K (zero-shot)
fc-clip	50.1	44.0	23.5
ConOVS (ours)	50.0	48.8	25.9

To investigate whether the proposed method remains effective under low-resource conditions, we conducted an additional experiment using a small incremental dataset. Specifically, we randomly sampled 5% of the Cityscapes training set and used it as the incremental dataset.

As shown in Table A12, our method maintains performance on the pre-training dataset while improving results on both the incremental and zero-shot datasets, even with the reduced incremental data. These findings demonstrate that ConOVS remains effective in scenarios where the number of incremental samples is limited.

G.6 Comparison with SemLA

Compared to SemLA [28], our approach shares a conceptual similarity in that both methods compute the proximity of an input image to each training dataset and use it to determine dynamic merging weights during inference. However, our method differs in three key design choices: (1) it uses both

image and text embeddings to assess domain relevance, whereas SemLA relies solely on image embeddings; (2) it estimates domain proximity using multivariate normal (MVN) distributions, while SemLA computes L2 distances to dataset-specific centroids; and (3) it fine-tunes the decoder for each dataset, in contrast to SemLA, which applies LoRA modules to the CLIP image encoder.

To provide a direct comparison, we re-implemented SemLA within our experimental framework. In this variant, domain proximity was computed only from image embeddings, estimated by calculating the L2 distance between the input embedding and dataset-specific centroids, and model adaptation was carried out through LoRA fine-tuning on the CLIP image encoder instead of decoder fine-tuning.

Table A13: Performance comparison between fc-clip, SemLA, and ConOVS. The best performance for each dataset is highlighted in bold.

Method	COCO (pre-training)	Cityscapes (incremental)	ADE20K (zero-shot)
fc-clip	50.1	44.0	23.5
SemLA	50.9	61.1	26.0
ConOVS (ours)	50.4	64.4	26.0

As shown in Table A13, our method (ConOVS) outperforms this SemLA variant on the incremental dataset. This improvement suggests that our design choices are better suited for continual open-vocabulary segmentation.

H Discussion

Applicability to Open-Vocabulary Object Detection (OVD). Our proposed method is not limited to Open-Vocabulary Segmentation (OVS) but can also be extended to OVD frameworks. Specifically, our approach is applicable to models with an encoder-decoder architecture that perform classification based on the similarity between image and text embeddings. This structural characteristic is shared by most OVD methods. For instance, YOLO-World [3] adopts a modular design consisting of an encoder and a prediction head, and omits the use of a conventional fc-based classifier. Given this structure, all components of our method can be directly incorporated without modification.

To be more specific, for each newly introduced training dataset, an MVN distribution can be formed using the encoder’s embeddings. During inference, the interpolation weights can be dynamically adjusted based on the proximity between the input sample and each domain, enabling the model to incrementally extend its recognition capability.

Effectiveness in Weakly Supervised Settings. We believe that ConOVS can function effectively regardless of the issue of low-quality segmentation annotations in weakly labeled datasets [10–12]. This is because our technique constructs the MVN distribution using only image and text embeddings, without relying on segmentation annotations. Therefore, we see no constraints in forming the MVN distribution and expect to accurately estimate the interpolation coefficients. While label noise may affect the overall performance, we contend that our method can still contribute to effectively enhancing the model’s recognition ability even in weakly labeled environments.

Applicability to Cost-based OVS Methods. Our method is also applicable to cost-based OVS approaches [4]. Since it is designed to merge independently trained models, it remains compatible with techniques that fine-tune the CLIP encoder. While cost-based methods typically generate segmentation maps by post-processing encoder features, our method does not rely on such steps, making it directly applicable without modification.

However, when applied to cost-based OVS models, the encoding process must be executed twice. This is because our method computes the interpolation factor based on the proximity of the input sample, which requires an initial forward pass through the encoder. After the interpolation factor is determined, a second forward pass is performed using the merged encoder to generate the final feature representation. Consequently, an increase in inference time is expected.

I Qualitative Results

This section presents a qualitative analysis of the original fc-clip, the standard fine-tuning technique, and ConOVS (ours). Figure 13 illustrates the qualitative outputs of each method. On the pre-training

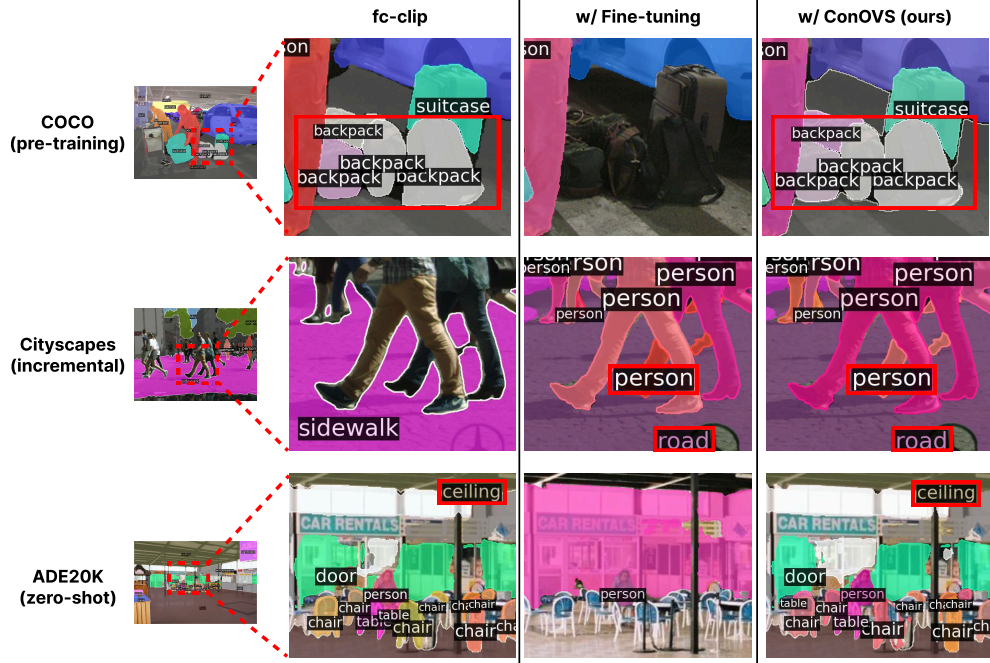


Figure I3: We provide a qualitative analysis on COCO, Cityscapes, and ADE20K. The comparison includes three methods: fc-clip, fine-tuning, and ConOVS (ours). Cityscapes is used as the incremental dataset.

dataset, the fine-tuning technique fails to recognize the *backpack*, indicating a loss of information from the pre-training stage. On the incremental dataset, fc-clip fails to identify key elements such as *road* and *person*, highlighting that OVS models perform well only within the distribution of the pre-training dataset. On the zero-shot dataset, the fine-tuning technique fails to recognize *ceiling*, a class absent from both the pre-training and incremental dataset. In contrast, the proposed method successfully identifies both previously learned and newly introduced classes, as well as classes not included in either training dataset.

J Results of All Evaluation Metrics

In this study, we consider fc-clip and X-Decoder as baseline models, both of which are designed to handle panoptic, instance, and semantic segmentation tasks. Based on this capability, we assess their performance using the corresponding metrics: PQ, mAP, and mIoU. As shown in Tables A14 to A16, the proposed method consistently outperforms all other approaches across the pre-training, incremental, and zero-shot test datasets, regardless of the evaluation metric. This confirms, as noted in Section A.3, that the three metrics follow similar performance trends. Moreover, the results verify that the proposed method performs robustly across all segmentation tasks.

Table A14: Performance comparison among the baseline, fine-tuning, retraining, existing continual learning methods, and ConOVS (ours). The incremental dataset is Cityscapes. We use PQ, mAP, and mIoU as evaluation metrics.

Method	Incremental Dataset	COCO (pre-training)				Cityscapes (incremental)				ADE20K (zero-shot)			
		PQ	mAP	mIoU	Avg	PQ	mAP	mIoU	Avg	PQ	mAP	mIoU	Avg
fc-clip	-	50.1	41.1	52.0	47.7	44.0	26.8	56.2	42.4	23.5	17.1	30.4	23.7
Fine-tuning	Cityscapes	-22.7	-16.2	-11.8	-16.9	+20.1	+13.9	+21.2	+18.4	-10.3	-6.3	-3.9	-6.8
Retraining		+0.6	+0.9	+0.3	+0.6	+17.9	+10.4	+18.7	+15.7	+1.7	-1.5	+2.5	+0.9
ER		-1.6	-2.7	+0.2	-1.4	+19.0	+13.0	+20.1	+17.4	+0.3	-3.5	+0.9	-0.8
LwF		-10.7	-11.9	-7.9	-10.2	+12.2	+2.7	+10.2	+8.3	-0.8	-5.4	+0.8	-1.8
EWC		-25.9	-19.0	-13.3	-19.4	+19.3	+11.2	+18.4	+16.3	-9.8	-8.4	-4.2	-7.5
ECLIPSE		-6.0	-6.2	-3.9	-5.3	+2.2	+0.2	+4.3	+2.2	+0.9	-3.6	+2.0	-0.3
ConOVS (ours)		+0.3	+0.5	+0.1	+0.3	+20.2	+13.9	+21.3	+18.5	+2.5	-1.2	+2.5	+1.3
X-Decoder	-	56.7	46.9	67.4	57.0	36.3	25.4	52.9	38.2	16.7	11.7	24.9	17.8
Fine-tuning	Cityscapes	-50.4	-32.2	-53.7	-45.5	+26.6	+11.7	+26.7	+21.7	-12.9	-8.1	-19.7	-13.5
ConOVS (ours)		-0.4	-0.4	-0.3	-0.3	+26.6	+11.6	+26.7	+21.7	+0.1	+0.5	-0.3	+0.1

Table A15: Performance comparison among the baseline, fine-tuning, retraining, existing continual learning methods, and ConOVS (ours). The incremental dataset is ADE20K. We use PQ, mAP, and mIoU as evaluation metrics.

Method	Incremental Dataset	COCO (pre-training)				ADE20k (incremental)				Cityscapes (zero-shot)			
		PQ	mAP	mIoU	Avg	PQ	mAP	mIoU	Avg	PQ	mAP	mIoU	Avg
fc-clip	-	50.1	41.1	52.0	47.7	23.5	17.1	30.4	23.7	44.0	26.8	56.2	42.4
Fine-tuning	ADE20K	-7.7	-6.2	-2.7	-5.5	+24.1	+19.0	+22.0	+21.7	-3.0	-2.8	+2.9	-1.0
Retraining		+1.4	+2.2	+2.9	+2.2	+16.5	+13.3	+15.0	+14.9	-1.2	-0.7	+1.3	-0.2
ER		+0.4	-0.3	+2.9	+1.0	+21.5	+16.3	+19.5	+19.1	-3.5	-2.8	-1.0	-2.4
LwF		-3.8	-7.1	-2.4	-4.4	+13.7	+8.4	+11.3	+11.1	-1.0	-6.2	-3.0	-3.4
EWC		-11.1	-9.3	-6.0	-8.8	+20.7	+16.2	+18.0	+18.3	-2.6	-3.2	+0.3	-1.8
ECLIPSE		-0.5	-1.2	+0.6	-0.3	+0.2	-0.3	+3.0	+1.0	-5.9	-4.0	-2.2	-4.0
ConOVS (ours)		+1.7	+1.4	+3.2	+2.1	+23.8	+18.6	+21.1	+21.2	+0.8	+0.8	+1.8	+1.1
X-Decoder	-	56.7	46.9	67.4	57.0	16.7	11.7	24.9	17.8	36.3	25.4	52.9	38.2
Fine-tuning	ADE20K	-37.3	-33.6	-42.4	-37.8	+28.2	+18.6	+27.2	+24.6	-3.7	-9.4	-0.8	-4.6
ConOVS (ours)		-1.5	-1.7	-1.1	-1.4	+29.2	+19.0	+27.5	+25.2	+1.4	-6.4	+3.5	-0.5

Table A16: Performance comparison among the baseline, fine-tuning, retraining, existing continual learning methods, and ConOVS (ours). The underlined values indicate the best score for each dataset. We use PQ, mAP, and mIoU as evaluation metrics.

Method	Learning Sequence	COCO (pre-training)			ADE20k (incremental)			Cityscapes (incremental)		
		PQ	mAP	mIoU	PQ	mAP	mIoU	PQ	mAP	mIoU
fc-clip	-	50.1	41.1	52.0	23.5	17.1	30.4	44.0	26.8	56.2
Fine-tuning	ADE20k → Cityscapes	20.8	19.5	40.0	15.4	14.2	34.9	<u>65.2</u>	<u>42.3</u>	<u>77.6</u>
Fine-tuning	Cityscapes → ADE20k	39.3	32.4	48.3	<u>48.3</u>	<u>36.3</u>	<u>52.1</u>	46.0	26.4	61.5
Retraining	COCO, Cityscapes, ADE20k	48.6	40.5	50.9	35.5	25.8	38.3	60.5	36.5	74.3
ConOVS (ours)	Cityscapes, ADE20k	<u>51.6</u>	<u>42.5</u>	<u>55.3</u>	47.0	35.9	51.4	64.3	40.7	<u>77.6</u>

References

- [1] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.
- [2] Zhiyuan Chen and Bing Liu. *Lifelong machine learning*. Springer Nature, 2022.
- [3] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16901–16911, 2024.
- [4] Seokju Cho, Heeseong Shin, Sunghwan Hong, Anurag Arnab, Paul Hongsuck Seo, and Seungryong Kim. Cat-seg: Cost aggregation for open-vocabulary semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4113–4123, 2024.
- [5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.
- [7] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.
- [8] Martin Hahner, Dengxin Dai, Christos Sakaridis, Jan-Nico Zaeche, and Luc Van Gool. Semantic understanding of foggy scenes with purely synthetic data. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3675–3681. IEEE, 2019.
- [9] Haiyang Huang, Newsha Ardalani, Anna Sun, Liu Ke, Shruti Bhosale, Hsien-Hsin Lee, Carole-Jean Wu, and Benjamin Lee. Toward efficient inference for mixture of experts. *Advances in Neural Information Processing Systems*, 37:84033–84059, 2024.
- [10] Dongjun Hwang, Jung-Woo Ha, Hyunjung Shim, and Junsuk Choe. Entropy regularization for weakly supervised object localization. *Pattern Recognition Letters*, 169:1–7, 2023.
- [11] Dongjun Hwang, Hyoseo Kim, Doyeol Baek, Hyunbin Kim, Inhye Kye, and Junsuk Choe. Curriculum learning with class-label composition for weakly supervised semantic segmentation. *Pattern Recognition Letters*, 188:171–177, 2025.
- [12] Dongjun Hwang, Seong Joon Oh, and Junsuk Choe. Small object matters in weakly supervised object localization. *Neurocomputing*, page 130494, 2025.
- [13] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.
- [14] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 699–715. Springer, 2020.
- [15] Rawal Khirodkar, Brandon Smith, Siddhartha Chandra, Amit Agrawal, and Antonio Criminisi. Sequential ensembling for semantic segmentation. *arXiv preprint arXiv:2210.05387*, 2022.
- [16] Beomyoung Kim, Joonsang Yu, and Sung Ju Hwang. Eclipse: Efficient continual learning in panoptic segmentation with visual prompt tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3346–3356, 2024.
- [17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [18] Minh Le, Huy Nguyen, Trang Nguyen, Trang Pham, Linh Ngo, Nhat Ho, et al. Mixture of experts meets prompt-based continual learning. *Advances in Neural Information Processing Systems*, 37:119025–119062, 2024.

- [19] Minhyeok Lee, Suhwan Cho, Jungho Lee, Sunghun Yang, Heeseung Choi, Ig-Jae Kim, and Sangyoun Lee. Effective sam combination for open-vocabulary semantic segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26081–26090, 2025.
- [20] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- [21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
- [22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.
- [23] David Lopez-Paz and Marc’ Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.
- [24] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [25] Martin Mundt, Yongwon Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336, 2023.
- [26] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Buló, and Peter Kotschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017.
- [27] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.
- [28] Reza Qorbani, Gianluca Villani, Theodoros Panagiotakopoulos, Marc Botet Colomer, Linus Härenstam-Nielsen, Mattia Segu, Pier Luigi Dovesi, Jussi Karlgren, Daniel Cremers, Federico Tombari, et al. Semantic library adaptation: Lora retrieval and fusion for open-vocabulary semantic segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9804–9815, 2025.
- [29] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14*, pages 102–118. Springer, 2016.
- [30] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7374–7383, 2019.
- [31] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [32] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.
- [33] Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks. *Advances in Neural Information Processing Systems*, 35:4203–4217, 2022.
- [34] 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*, pages 2636–2645, 2020.
- [35] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.
- [36] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019.