EAGLE: Eigen Aggregation Learning for Object-centric Unsupervised Semantic Segmentation (Supplementary Material)

Contents

A. Additional Material: Project Page & Presentation Video

We have described our results in an easily accessible manner on our project page, where a brief **presentation video** is also available. The link to the **project page** is as follows: https://micv-yonsei.github.io/eagle2024/.

B. Additional Evaluation Results

In this section, we extend our discussion to include the evaluation results of *EAGLE*. Initially, we present both quantitative and qualitative findings from the **Potsdam-3** dataset [9], which were not covered in the main paper due to space constraints. Subsequently, we also provide additional qualitative analysis of the COCO-Stuff [1] and Cityscapes datasets [4].

Table 1. Quantitative results on Potsdam-3 dataset [9].

Method	Backbone	Unsup. Acc.	Unsup. mIoU
Random CNN [9]	VGG11	38.2	-
K-means [13]	VGG11	45.7	-
SIFT [11]	VGG11	38.2	-
ContextPrediction [5]	VGG11	49.6	-
CC [8]	VGG11	63.9	-
DeepCluster [2]	VGG11	41.7	-
IIC [9]	VGG11	65.1	-
DINO [3]	ViT-B/8	53.0	-
+ STEGO [7]	ViT-B/8	77.0	62.6
+ HP [14]	ViT-B/8	82.4	68.6
+ *EAGLE (Ours)*	ViT-B/8	**83.3**	**71.1**

B.1. Potsdam-3

In Table 1, we present the quantitative results for the Potsdam-3 dataset [9], where our *EAGLE* sets a new score. We not only report the unsupervised accuracy, as previously done by methods [7, 14], but also expand our reporting to include unsupervised mIoU. With the ViT-B/8 backbone, *EAGLE* surpass existing USS methods in unsupervised accuracy, with gains of **+6.3** over STEGO [7] and **+0.9** over HP [14]. In the context of unsupervised mIoU, our method surpasses STEGO by a significant margin of **+8.5**, and **+2.5** over HP.

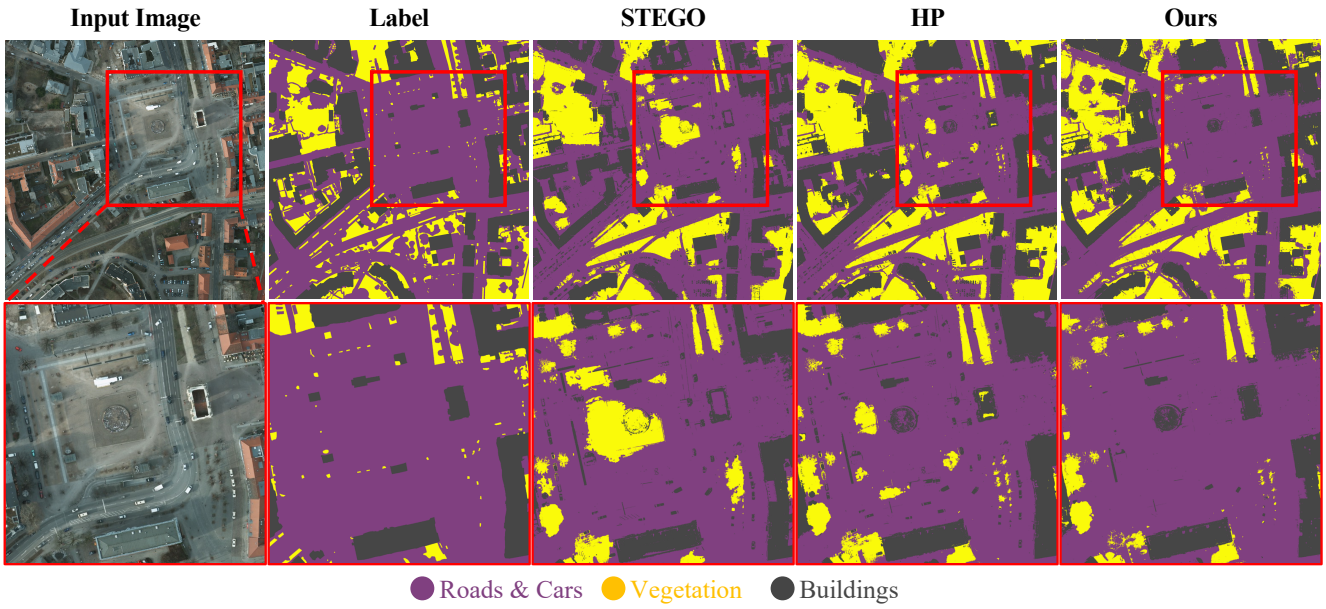

Figure 1. Qualitative results of Potsdam-3 dataset [9] trained with ViT-B/8 backbone.

In our qualitative analysis in Fig. 1, compared to STEGO [7] and HP [14], our *EAGLE* demonstrates a more accurate understanding of object-level semantics. Specifically, in the second row, which is a zoomed-in view of the red box in the first row, *EAGLE* successfully classifies the cars on the road as separate entities from the buildings. This distinction is not as clear in the results from STEGO and HP, highlighting the superior capability of our approach in discerning and segmenting objects according to their semantic categories.

B.2. COCO-Stuff

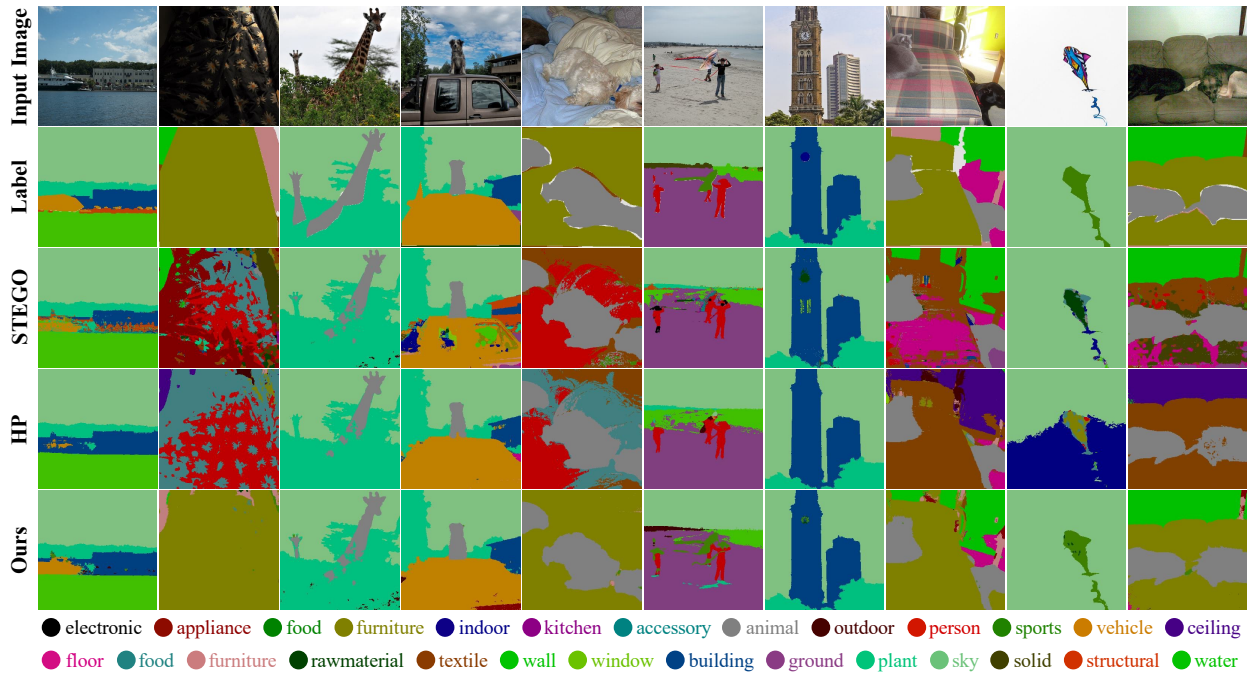

Figure 2. Additional qualitative results of COCO-Stuff dataset [1] trained with ViT-S/8 backbone.

B.3. Cityscapes

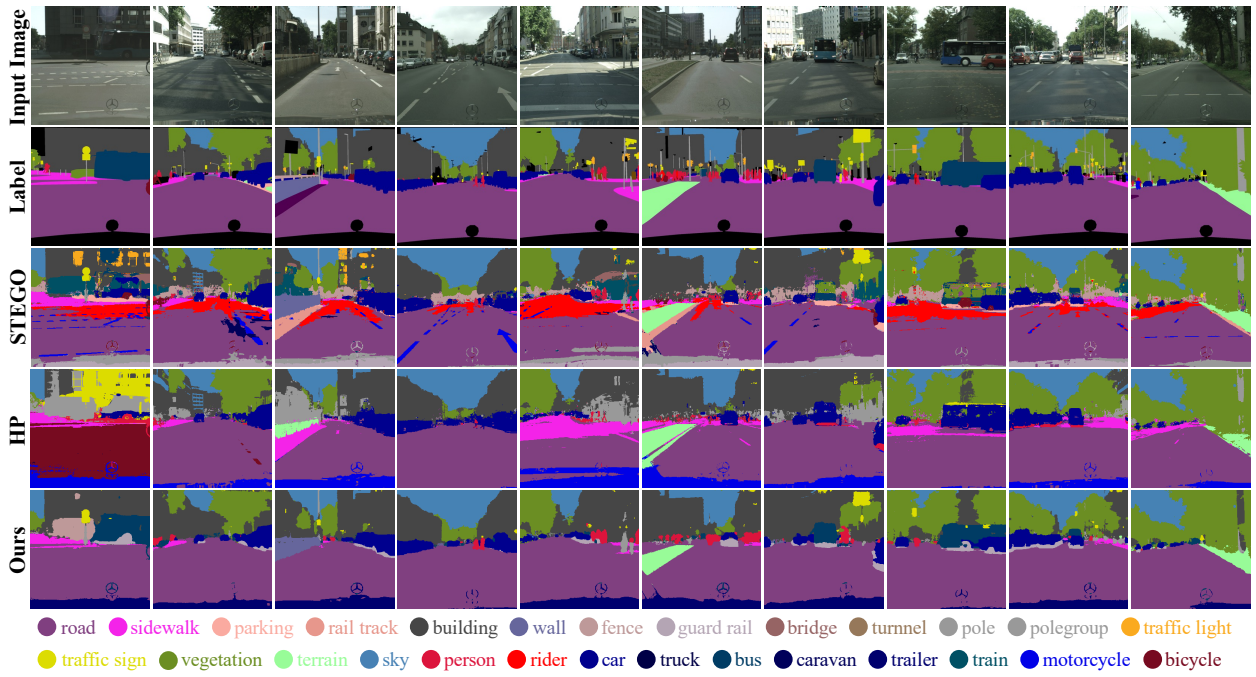

Figure 3. Additional qualitative results of Cityscapes dataset [4] trained with ViT-B/8 backbone.

C. Additional Experiments

C.1. Additional Ablation Study

In this section, we provide additional ablation analysis on the feature type (Section C.1.1) and prototype selection method (Section C.1.2).

C.1.1 Ablation: Feature Type

Table 2. The experimental results for the feature type on the COCO-Stuff dataset.

Feature Type		Unsupervised		Linear	
$\mathcal{S}_\theta$	$\mathbf{A}_{\text{seg}}$	Acc.	mIoU	Acc.	mIoU
$\mathbf{F}$	$\mathbf{F}$	43.1	17.1	74.1	41.2
	$\mathbf{K}$	57.6	24.9	74.6	41.6
	$\mathcal{S}_\theta(\mathbf{F})$	59.1	25.4	74.5	41.5
$\mathbf{K}$	$\mathbf{F}$	58.6	26.1	74.7	41.7
	$\mathbf{K}$	56.9	23.8	74.6	41.6
	$\mathcal{S}_\theta(\mathbf{K})$	**64.2**	**27.2**	**76.8**	**43.9**

In this section, we explore various different combinations of feature types that will be used for computing $\mathcal{S}_\theta$ and for the creation of $\mathbf{A}_{\text{seg}}$, thereby exploring their implications and potential applications in the context of our study. In Table 2, we analyze the experimental results for the COCO-Stuff dataset [1] trained using the ViT-S/8 backbone. The $\mathbf{F}$ and $\mathbf{K}$ listed under the $\mathcal{S}_\theta$ in the "Feature Type" column, represent the types of features inputted into the segmentation head $\mathcal{S}_\theta$. In this study, $\mathbf{F}$ is sourced from the activation map at the final layer of the vision transformer, while $\mathbf{K}$ is processed in accordance with the methods outlined in the main manuscript (Section 3.1). The $\mathbf{A}_{\text{seg}}$ located in the "Feature Type" column, denotes the feature types utilized in the creation of $\mathbf{A}_{\text{seg}}$. As detailed in the manuscript, EiCue construction involves the sum of two adjacency matrices to form the Laplacian, one of which is the semantic similarity matrix $\mathbf{A}_{\text{seg}}$, providing the semantic interpretation of the object. The $\mathbf{F}$ and $\mathbf{K}$ in the $\mathbf{A}_{\text{seg}}$ column, retain the same values as previously described, being sourced from a static pretrained vision transformer. When considering $\mathcal{S}_\theta(\mathbf{F})$ and $\mathcal{S}_\theta(\mathbf{K})$, these refer to the $\mathbf{F}$ and $\mathbf{K}$ features that have been processed through the segmentation head $\mathcal{S}_\theta$. As the model training progresses, these dynamic values are subject to change, reflecting the evolving state of the trained segmentation head.

As detailed in Table 2, we present the results for all possible feature type combinations. Overall, leveraging $\mathbf{K}$ to compute $\mathbf{S}$ through $\mathcal{S}_\theta$, yielded superior outcomes compared to utilizing $\mathbf{F}$. To construct $\mathbf{A}_{\text{seg}}$, utilizing dynamic feature types such as $\mathcal{S}_\theta(\mathbf{F})$ or $\mathcal{S}_\theta(\mathbf{K})$ demonstrates significantly higher values in contrast to the aforementioned static features, $\mathbf{F}$ and $\mathbf{K}$, thereby validating the effectiveness of our training approach. Ultimately, employing the learnable feature $\mathcal{S}_\theta(\mathbf{K})$ for $\mathbf{A}_{\text{seg}}$ delivered the best performance.

C.1.2 Ablation: Prototype Selection Method

Table 3. The experimental results for the prototype selection method on the COCO-Stuff dataset.

Method	Unsupervised		Linear	
	Acc.	mIoU	Acc.	mIoU
PCA	55.7	18.3	74.7	39.6
Centroid	59.0	24.9	75.8	42.1
Medoid	**64.2**	**27.2**	**76.8**	**43.9**

We further carry out ablation experiments comparing various methods for selecting prototypes. A prototype is a semantic vector that represents a single object, serving as an anchor that attracts semantic vectors within the object and repels the other ones. To select a semantic vector that represents an object, we can consider several options for choosing object-representative semantic vectors. **(a)** Principal Component Analysis (PCA): we can use PCA to find the direction of maximum variance in the data, and choose the vector that has the largest projection on the first principal component. **(b)** Centroid: calculates the mean vector of the set, where each component of the centroid is the average of that component across all vectors in the set.

(c) Medoid: we can choose the vector that minimizes the sum of distances to all other vectors. This is less sensitive to outliers compared to the centroid. As reported through Table 3, we observed that leveraging the Medoid method demonstrated the best performance.

C.2. Additional Visualization of the Primary Elements of Eigen Aggregation Module

C.2.1 Eigenvectors

Figure 4. Additional visualization of the eigenvector obtained by training the COCO-Stuff dataset [1] with ViT-S/8 as the backbone.

As illustrated in Fig. 4, we provide additional visualization of eigenvectors obtained from learnable feature $\mathbf{S}$ trained using

COCO-Stuff dataset [1]. Our eigenvectors present remarkable capability in distinguishing objects while capturing within its object semantics.

C.2.2 EiCue

Figure 5. Additional comparison between K-means and our EiCue, with EiCue demonstrating enhanced performance in discerning object semantics and structures relative to K-means.

As shown in Fig. 5, we present additional visualizations that compare our EiCue model with the traditional K-means clustering approach. EiCue shows a significant improvement over clustering with K-means in identifying the semantic details

of objects and discerning the comprehensive structure of images. This distinction is particularly evident in the way EiCue captures intricate object semantics and delineates the structural elements within the images. These observations substantiate our model's claim that EiCue is proficient in recognizing object semantics and distinguishing structural components.

C.3. Application

Our *EAGLE* is designed for semantic segmentation, which predicts dense class prediction. Consequently, *EAGLE* is applicable to a variety of tasks that necessitate pixel-level semantic interpretation. A key technique in our approach is the *eigendecomposition* of the Laplacian matrix. Thus, we can use eigenvectors of images and these eigenvectors are able to capture the detailed semantic structures present in an image. As discussed in Section C.3.1, leveraging these eigenvectors allows us to precisely perform image matting and achieve localized image stylization.

C.3.1 Image Matting

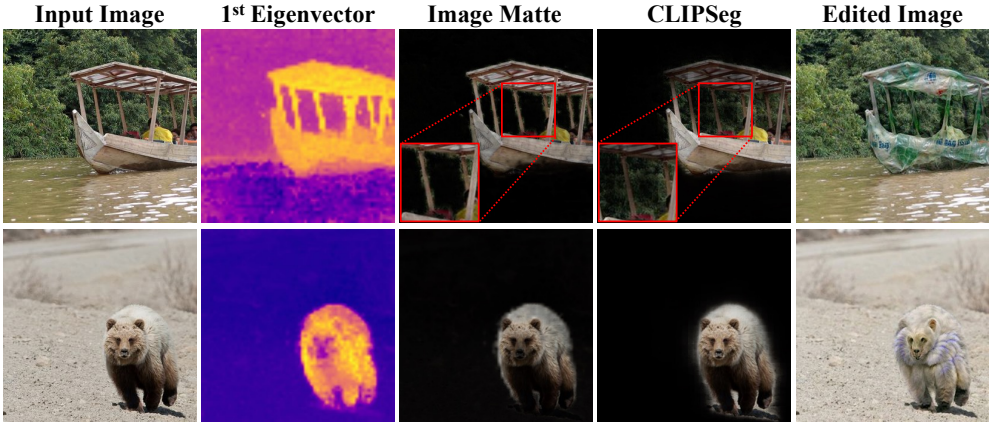

Figure 6. We demonstrate our Laplacian matrix-based image matting. From left to right: the input image, eigenvector from our matrix, resultant image matte, CLIP-based segmentation [12], and edited image using our matte with existing text-driven image editing model [10]. For the editing in the last column, the text prompts used were `plastic bag` for the top image and `white fur` for the bottom image. Our matte offers clearer object boundaries than CLIPSeg, leading to superior editing quality.

The eigenvectors obtained through our proposed method are effective in distinguishing objects within images. To this end, our eigenvectors can be efficiently utilized for image matting and localized image stylization. The process of image matting involves the extraction of the foreground from an image, facilitating further manipulations like compositing onto a different background or selective editing. Historically, the matting task has been challenging due to intricate object edges and subtle transitions. The eigenvectors, adept at distinguishing distinct objects or features within images, provide a powerful solution to this challenge. Using our proposed method, we leverage the potential of these eigenvectors for refined image matting. As depicted in Fig. 6, the first column represents the input images. The subsequent column showcases the first eigenvector derived from our matrix, effectively highlighting the structure of the primary object. The third column portrays the resultant image matte, distinctly separating the object from its surroundings. In contrast, the fourth column, representing the CLIP-based segmentation, while reasonable, but fails to provide as delicate boundaries as our eigenvector-based technique. A notable difference can be observed in the image of the `boat`. Our method adeptly separates the pillar of the boat, whereas the CLIP-based approach fails to isolate it, erroneously including the trees in the background as part of the foreground. The final column presents the edited images, emphasizing the utility of our matte for selective edits, ensuring that the distinguished object can be stylized without affecting the background.

D. Implementation Details

In this section, we discuss details of correspondence distillation loss (Section D.1), model architecture (Section D.2), and hyperparameters (Section D.3).

D.1. Correspondence Distillation Loss

By employing a correspondence distillation loss [7], we enhanced the stability of the training process by ensuring reliable graph Laplacian initialization. The original correspondence distillation loss is defined as

$$\mathcal{L}_{\text{cd}}(\check{\mathbf{F}}, \check{\mathbf{S}}, b) = -\sum \left(\check{\mathbf{F}} - b\right) \check{\mathbf{S}}, \tag{1}$$

where $\check{\mathbf{F}}$ and $\check{\mathbf{S}}$ is computed as a cosine distance using $\mathbf{F}$ and $\mathbf{S}$. Here, $\mathbf{F}$ is a feature obtained from the activation map at the final layer of the vision transformer with a given image and $\mathbf{S}$ is the projection of $\mathbf{F}$ using segmentation head $\mathcal{S}_\theta$. Since we leverage attention keys in place of $\mathbf{F}$, we substitute $\mathbf{F}$ with $\mathbf{K}$ and revise $\mathbf{S}$ to be $\mathbf{S} = \mathcal{S}_\theta(\mathbf{K})$. Within the framework of existing correspondence distillation loss [7], which involves three distinct loss functions, our method modifies and utilizes two of these components: **(a)** the augmented image correspondence distillation loss and the **(b)** random image correspondence distillation loss.

Although Eq. (1) is applied to both types of loss, the difference lies in what each correspondence tensor represents. **(a)** In the augmented image correspondence distillation loss, $\check{\mathbf{K}}_{\text{aug}}$ and $\check{\mathbf{S}}_{\text{aug}}$ is computed as a cosine distance between $\mathbf{K}, \tilde{\mathbf{K}}$ and $\mathbf{S}, \tilde{\mathbf{S}}$, respectively. $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{S}}$ are the results for the $\tilde{\mathbf{x}}$, which is the augmented images of $\mathbf{x}$, created through the same aforementioned process. While **(b)** in the random image correspondence distillation loss, $\check{\mathbf{K}}_{\text{rand}}$ and $\check{\mathbf{S}}_{\text{rand}}$ is computed as a cosine distance between $\mathbf{K}, \acute{\mathbf{K}}$ and $\mathbf{S}, \acute{\mathbf{S}}$, respectively. $\acute{\mathbf{K}}$ and $\acute{\mathbf{S}}$ are the results for the random images from the entire dataset, created through the same aforementioned process. In the Eq. (1), b is defined as the *shift* of the feature value and remained fixed throughout the training process. In contrast, we modified b_{aug} and b_{rand} to dynamically adapt based on the $\check{\mathbf{K}}$ and $\check{\mathbf{S}}$ in both losses, where b_{aug} represents the b in augmented image correspondence distillation loss, and b_{rand} represents the b in random image correspondence distillation loss. Here is the formula we used:

$$b_{\text{aug}} = \left| \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \check{\mathbf{K}}_{ij} - \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \check{\mathbf{S}}_{ij} - k_{\text{shift}} \right|, \tag{2}$$

$$b_{\text{rand}} = \left(\frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \check{\mathbf{K}}_{ij} + \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \check{\mathbf{S}}_{ij} - k_{\text{shift}} \right) \times v_{\text{shift}}, \tag{3}$$

where H and W refer to the height and width of the feature tensor, respectively. Here, k_{shift} and v_{shift} are determined as hyperparameters (see Section D.3) Thus, the final correspondence distillation loss that we use is defined as

$$\mathcal{L}_{\text{corr}} = \mathcal{L}_{\text{cd}}(\check{\mathbf{K}}_{\text{aug}}, \check{\mathbf{S}}_{\text{aug}}, b_{\text{aug}}) + \mathcal{L}_{\text{cd}}(\check{\mathbf{K}}_{\text{rand}}, \check{\mathbf{S}}_{\text{rand}}, b_{\text{rand}}), \tag{4}$$

which is the summation of augmented image correspondence distillation loss and random image correspondence distillation loss.

D.2. Detailed Architecture

Image Encoder. For all experiments, we basically leverage DINO [3] pretrained ViT-S/8, ViT-S/16 and ViT-B/8 [6] as an image encoder. Specifically, we initialize ViT with a teacher weight of DINO. As mentioned before, we extract attention keys hierarchically from ViT and then concatenate them into a single feature tensor $\mathbf{K}$ (for details, see Section 3.1 in the main manuscript). Then, we apply channel-wise dropout ($p = 0.1$) to feature tensor $\mathbf{K}$ before feeding to segmentation head $\mathcal{S}_\theta$.

Segmentation Head. We illustrate the detailed architecture of the segmentation head and projection head in Fig. 7. For a fair performance comparison with existing models, we employ the same approach for the segmentation head $\mathcal{S}_\theta$ as used in the previous models [7, 14]. This non-linear segmentation head $\mathcal{S}_\theta$ consists of simple linear layers. The input is a tensor $\mathbf{K}$ with a dimension of D_K. This tensor first passes through a linear layer that transforms its dimension from D_K to D_S, where D_S represents the desired dimension for the output of $\mathcal{S}_\theta$. Following the initial linear transformation, there is a ReLU (Rectified Linear Unit) activation function, which introduces non-linearity to the process. The output of the activation layer is then fed into another linear layer, which once again maps the dimension from D_K to D_S. The outputs of the two pathways are then combined via a summation operation. The summation consolidates the linearly transformed input and the non-linearly transformed input. The result is the tensor $\mathbf{S}$ with the dimension D_S, which is the output of the segmentation head.

Projection Head. As shown in Fig. 7, we project semantic tensor $\mathbf{S}$ to $\mathbf{Z}$ to facilitate object-centric contrastive learning. The basic concept of projection head $\mathcal{Z}_\xi$ is to project the tensor without transforming its input dimension. Following this concept,

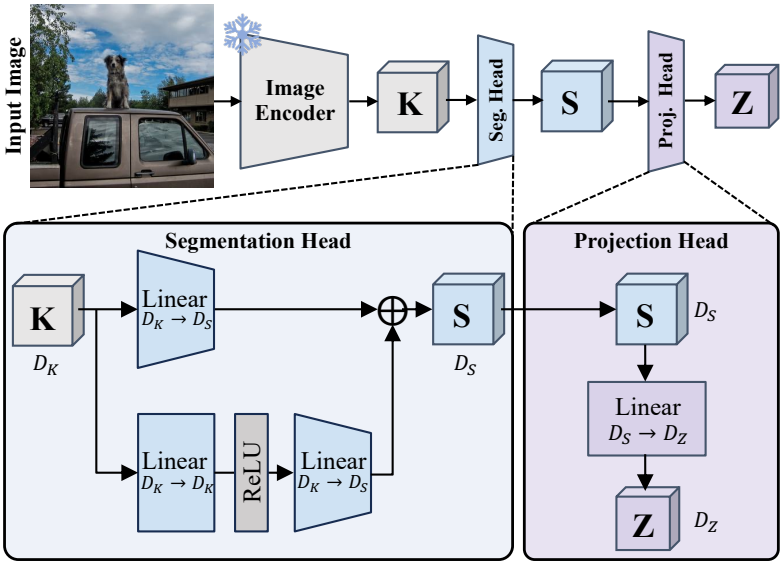

Figure 7. Detailed architecture of segmentation head and projection head used in our method.

we form a projection head with a single linear layer that maps its dimension from D_S to D_Z. Here, we note that while we use different notations D_S and D_Z for clarity and ease of explanation, the actual dimensions represented by these notations are the same as $D_S = D_Z$.

D.3. Hyperparameters

Table 4. Hyperparameters used in *EAGLE*. LR refers learning rate.

Hyperparams		COCO-Stuff ViT-S/8	Cityscapes ViT-B/8
λ_{obj}		0.3	0.3
λ_{sc}		0.7	0.7
λ_{nce}		0.9	0.7
k_{shift}		0	0.11
v_{shift}		3.5	3.5
step		200	380
LR	$\mathcal{S}_\theta, \mathcal{Z}_\xi, \Phi$	0.0005	0.0005
	$\mathbf{C}$	0.00005	0.0004

In this section, we carefully describe hyperparameters in Table 4 that are used throughout our series of experiments. In the table above, "step" refers to the number of training iterations required for the λ_{nce} to increase from 0 to the number indicated in λ_{nce} row. "LR" indicates learning rate and $\mathcal{S}_\theta, \mathcal{Z}_\xi, \Phi$ shares same learning rate.

E. Discussion

E.1. Failure Cases

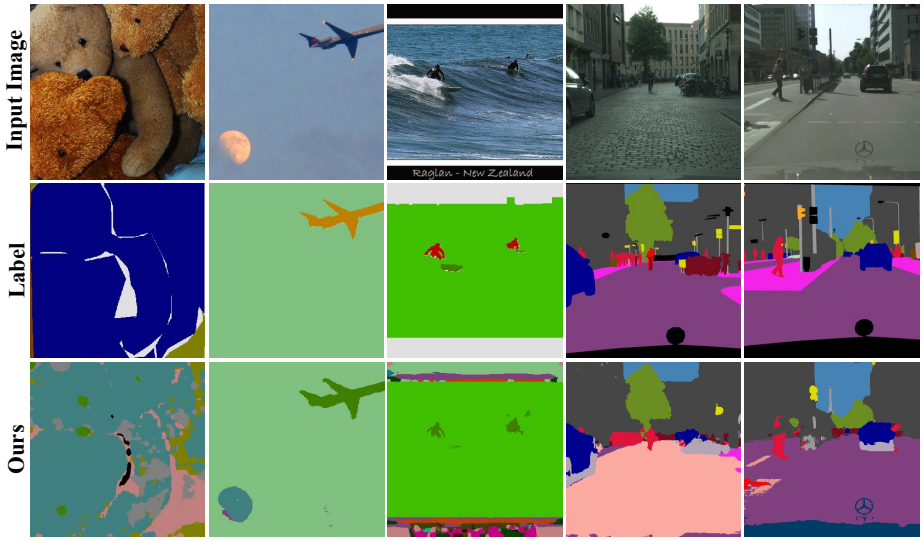

Figure 8. Failure cases of *EAGLE*.

Unsupervised semantic segmentation (USS), unlike the fully-supervised approach, is quite challenging as it predicts classes for each pixel without labeled data. While USS is likely to show much better performance in the future, our model represents a step in its evolution and thus comes with certain limitations. Fig. 8 illustrates the failure cases of EAGLE, which has been trained to capture object-level semantics. The first to the third column is from the COCO-Stuff dataset and the remaining columns are from the Cityscapes dataset. In the first column, our model fails to segment objects properly. This is due to the narrow color distribution of the input image and the limited variety of object semantics present in the image, leading to a failure in creating a high-quality adjacency matrix for EiCue. In the second and third columns, we observed that our results successfully implemented object-level semantics but made errors in matching the object class. Within column four, we see our results that accurately segment and correctly classify `car`, `tree`, `building`, and `person`, but incorrectly categorize gravel paths as a different class instead of `road`. Similarly, in the last column, there were no critical errors for objects other than `sidewalk`. However, even when viewed with the human eye, the input image presents a challenging scenario in distinguishing between `road` and `sidewalk`.

E.2. Future Works

Throughout our manuscript, we demonstrated that leveraging EiCue through graph Laplacian effectively captures the semantic structure of an image. However, constructing an adjacency matrix and forming a Laplacian matrix entails a relatively high computational cost. This approach does not affect the inference time in our framework, but it does require more training time compared to using solely deep-based methods. In our research, we compute the adjacency matrix for every feature vector of an image. While *EAGLE* shows state-of-the-art results, regarding every single feature vector is not the most computationally efficient, suggesting that improvements to EiCue could be made by sampling only vital features based on other knowledge of the image and constructing the Laplacian matrix accordingly. Additionally, as our primary focus is on object-level semantics, this approach may not be directly applicable to domains like medical imaging. Therefore, it is crucial to engage in research that uncovers knowledge about object-level semantics, which is applicable across multiple domains and holds significant potential for widespread use in the field of computer vision.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1209–1218, 2018. 1, 3, 4, 5, 6

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018. 2

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021. 2, 8

[4] 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. 1, 3

[5] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 2

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 8

[7] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T. Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *International Conference on Learning Representations*, 2022. 2, 8

[8] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015. 2

[9] Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019. 1, 2

[10] Seung Hyun Lee, Chanyoung Kim, Wonmin Byeon, Sang Ho Yoon, Jinkyu Kim, and Sangpil Kim. Lisa: Localized image stylization with audio via implicit neural representation. *arXiv preprint arXiv:2211.11381*, 2022. 7

[11] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision*, pages 1150–1157. Ieee, 1999. 2

[12] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, 2022. 7

[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 2

[14] Hyun Seok Seong, WonJun Moon, SuBeen Lee, and Jae-Pil Heo. Leveraging hidden positives for unsupervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19540–19549, 2023. 2, 8