# Bias-Resilient Weakly Supervised Semantic Segmentation Using Normalizing Flows — Supplementary Material

## A. More Details about Normalizing Flow

We mainly follow the DDFP [5], the normalizing flow network is designed as a modified RealNVP [2]. As shown in Fig. 1, the input feature $v$ in the original feature space is evenly divided into $v_1$ and $v_2$ at the channel level. After passing through a invertible neural network, $v_1$ performs Element-Wise multiplication with $v_2$, and then performs Element-Wise sum with the transformed $v_1$ to obtain $v_2'$. Finally, $v_2'$ and $v_1'$ are merged at the channel level to obtain the final normalizing flow mapping feature $z$. The invertible neural network contains two Linear layers with learnable parameters. The invertible neural network is implemented by FrEIA [1]. To the training stability of normalizing flow, we add small-scale random noise on features which is also be adapted in FlowGMM [3].

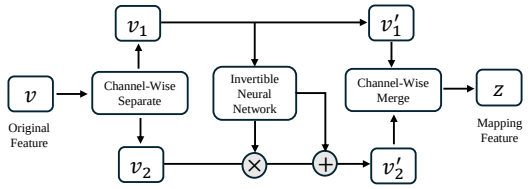


Figure 1. Network Architecture of Normalizing Flow.

## B. More Details about WSSS Settings

During WSSS training, images are randomly cropped to $512 \times 512$. For prototype CAM inference, we use multi-scale inference with image scale ratios of $\{0.5, 1.0, 1.5, 2.0\}$. The semantic segmentation model employed is DeepLab V2 with a ResNet101 backbone, pretrained on the ImageNet dataset. For PASCAL VOC 2012, we adhere to the default training settings of HSC [6], where input images are randomly scaled to $\{0.5, 1.5\}$ and cropped to $448 \times 448$ for training, with a batch size of 10 and 20k iterations. For MS COCO 2014, we follow the default training settings of CLIP-ES [4], scaling input images to $\{0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$ and cropping them to $481 \times 481$. The batch size is set to 5, with 100k iterations. All our experiments are conduct on RTX 3090 GPU.

| ($\tau_l$,$\tau_h$) | (0.3,0.9) | (0.3,0.8) | **(0.3,0.7)** | (0.3,0.6) | (0.3,0.5) |
|---|---|---|---|---|---|
| mIoU(%) | 69.9 | 70.8 | **72.1** | 72.1 | 71.9 |
| ($\tau_l$,$\tau_h$) | (0.3,0.4) | (0.4,0.7) | (0.5,0.7) | (0.6,0.7) | (0.7,0.9) |
| mIoU(%) | 71.8 | 72.1 | 72.0 | 71.9 | 71.4 |

Table 1. Sensitivity analysis on filtering threshold $\tau_l$ and $\tau_h$ in pseudo-label generation for modeling pixel feature distribution. The bold indicates the parameters we chose finally in our work.

| $\xi$ | 0.95 | **0.9** | 0.85 | 0.8 |
|---|---|---|---|---|
| mIoU(%) | 71.8 | **72.1** | 71.3 | 69.9 |

Table 2. Sensitivity analysis on background identification parameter $\xi$ in BSDNF. The bold indicates the parameters we chose finally in our work.

## C. Ablation study on Filtering Threshold

Tab. 1 shows the impact of filtering thresholds $\tau_l$ and $\tau_h$ on generating pseudo-labeled and unlabeled pixel features for training the normalizing flow. The results indicate that having enough labeled pixel features is crucial for performance. For instance, a high $\tau_h$ and low $\tau_l$ (*e.g.,* $\tau_l = 0.3$ and $\tau_h = 0.9$) lead to significant performance degradation due to limited labeled features and excessive unlabeled features, which destabilize the normalizing flow training. Conversely, setting $\tau_l$ too high (*e.g.,* $\tau_l = 0.6$ and $\tau_h = 0.7$) also causes some performance drop, highlighting the importance of introducing some unlabeled features to participate in the training of normalizing flow.

## D. Ablation study on BSDNF

Tab. 2 shows the impact of background identification parameter $\xi$ in BSDNF. We can find that setting $\xi$ as 0.9 can get the best performance. A higher $\xi$ (*e.g.,* 0.95) results in fewer recognized noises and insufficient noise suppression, which is an extremely conservative strategy that leads to marginal gains; And lower $\xi$ (*e.g.,* 0.85 and 0.8) means relaxing the background identification, which may result in some true foreground pixels being suppressed and leading to performance degradation.

## E. More visualization about ablations and failure cases

Fig. 2 (a) shows that CCNF largely reduces false activations. FSCL further mitigates semantic bias, while BSDNF eliminates false activations on the background. In Fig. 2 (b), BRNF fails to activate targets with scarce class patterns. It is because NF models rare class pixel features to the margin of distribution during training, hindering the model to capture such patterns.

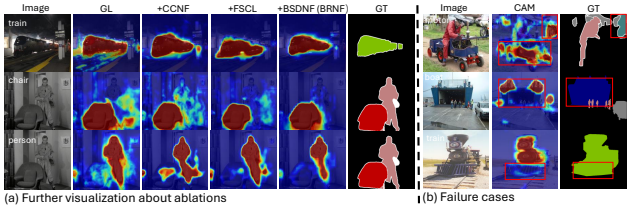


Figure 2. Ablation visualizations and failure cases.

## F. Discussions about Gaussian Mixture Model and Normalizing Flow

The normalized flow model offers several advantages over the Gaussian Mixture Model (GMM): **Flexibility in Distribution Modeling:** Normalizing Flow can capture more diverse and flexible distributions by adjusting the transformations applied to the data, allowing it to fit a wide variety of data shapes. But GMM assumes that data points are generated from a mixture of Gaussian distributions, which restricts the model's ability to fit data that doesn't adhere to this assumption. **Scalability:** Normalizing Flow supports fully online learning and can scale to large datasets, making it suitable for tasks requiring continuous learning and adaptation. But GMM generally trained using batch methods like Expectation-Maximization (EM), which can be less efficient and less scalable, especially for large or streaming datasets. **Handling of High-Dimensional Data:** Normalizing Flow is designed to handle high-dimensional data by using invertible transformations that maintain the structure of the data. But GMM can struggle with high-dimensional data, as fitting a mixture of Gaussians in such spaces can become computationally expensive and prone to overfitting or underfitting. **Exact Likelihood Estimation:** Normalizing Flow provides exact likelihood estimation for the data, which is useful for tasks where accurate probability estimates are critical. Although GMM also provides likelihood estimates, these are based on the assumption that the data follows a mixture of Gaussian distributions, which might not always hold true. **In summary, compared to Gaussian mixture model, normalizing Flow is more suitable for modeling pixel features of the entire dataset which are large-scale, complex, high-dimension and dynamic.**

## References


[1] Lynton Ardizzone, Till Bungert, Felix Draxler, Ullrich Köthe, Jakob Kruse, Robert Schmier, and Peter Sorrenson. Framework for easily invertible architectures (freia), 2018-2022. *URL https://github. com/vislearn/FrEIA*, 2018. 1

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1

[3] Pavel Izmailov, Polina Kirichenko, Marc Finzi, and Andrew Gordon Wilson. Semi-supervised learning with normalizing flows. In *International Conference on Machine Learning*, 2020. 1

[4] Yuqi Lin, Minghao Chen, Wenxiao Wang, Boxi Wu, Ke Li, Binbin Lin, Haifeng Liu, and Xiaofei He. Clip is also an efficient segmenter: A text-driven approach for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1

[5] Xiaoyang Wang, Huihui Bai, Limin Yu, Yao Zhao, and Jimin Xiao. Towards the uncharted: Density-descending feature perturbation for semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1

[6] Yuanchen Wu, Xiaoqiang Li, Songmin Dai, Jide Li, Tong Liu, and Shaorong Xie. Hierarchical semantic contrast for weakly supervised semantic segmentation. In *International Joint Conference on Artificial Intelligence*, 2023. 1