All-pairs Consistency Learning for Weakly Supervised Semantic Segmentation Supplementary Material

Weixuan Sun[1,2,3], Yanhao Zhang[5], Zhen Qin[2], Zheyuan Liu[1], , Lin Cheng[4], Fanyi Wang[5], Yiran Zhong[2,3], Nick Barnes[1]
[1]Australian National University, [2]OpenNLPLab, [3]Shanghai AI Lab,
[4]Xiamen University, [5]OPPO Research Institute.

1. Implementation Details

Our code is implemented in PyTorch on four NVIDIA RTX2080Ti GPUs. During classification training, we use ViT-hybrid-B [3] as the backbone. The training images are randomly resized and cropped to 384 $\times$ 384 and we use a batch size of 4. The model is trained for 15 epochs using the SGD optimizer with an initial learning rate of 0.01, weight decay of $5e-4$, and Polynomial Learning Rate Policy. In Equation 6, we set $\alpha = \beta = 100$.

Evaluation Metric and Protocol For class localization maps, we report the best mean Intersection-over-Union (mIoU), i.e., the best match between the activation maps and the segmentation ground truth under all background thresholds. For semantic segmentation (in mIoU), we obtain the PASCAL VOC *val* and MS COCO results by comparing the predictions with their ground truth, while we obtain the PASCAL VOC *test* results from the PASCAL VOC online evaluation server.

2. Spatial Transformation and Inversion

As discussed in the main paper Section 3.2, our consistency regularization requires an inverse transformation f^{-1}, which restores the spatial ordering of the tokens within the transformer. This is needed as the augmentation changes the pixel orderings in the spatial domain, which in turn alters the orderings of the patches, hence, tokens within the transformer (Fig. 1). Restoring the order is therefore necessary for us to match the corresponding patches *before and after* the augmentation, so that we can compute losses.

Here, we present a toy example to demonstrate such an effect. As shown in Fig. 1 (top), we have an input image of resolution 4×4 with unique number for each pixel. We then process it with a patch size of 2×2, with each patch in a different color. Then, we flatten the patches into a 1-d sequence of tokens as the input of the vision transformer. In Fig. 1 (bottom), we horizontally flip the image. Likewise,

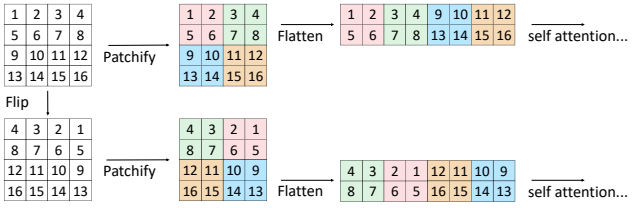

Figure 1. Concept illustration of spatial transformation. The spatial augmentation not only transforms patch orders but also the content inside each patch. We propose transformation inversion to invert the patches to the original order.

the augmented view is converted into patches that are flattened as a 1-d sequence of tokens. By comparing the colored blocks, a token-wise correspondence is easily drawn and order is restored. We point out that *internally*, the embedding of a token is still different than its corresponding counterpart due to the augmentation applied in the pixel domain. But still, they shall share a similar activation signal as they are obtained from the same entity — albeit flipped or rotated — and our aim is to regularize such signals through losses.

To compute the losses for regularization, we require comparing elements *inside* the self-attention matrices of the transformer, which means we need to restore the orderings of the said elements in a similar philosophy as what shown in Fig. 1. Though, this is far less trivial, as self-attention is calculated among all tokens. Below, we provide detailed derivations on how we obtain the inverse transformation f^{-1} that restores such orderings.

2.1. Derivation of the Transformation Inversion

We consider the transformation inversion of the token ordering in this section. By transformation inversion, we ensure the augmented attention matrix has an equal spatial ordering to the original attention matrix. Note that we only consider the token ordering in this section and omit the transformation that has been applied inside each patch

of the image as we only aim to restore the original spatial information, not the embedding.

Notations and Lemmas First, we define a general operation $\mathbf{vec}(\cdot)$ which converts an arbitrary 2-d vector (i.e., feature map) into a 1-d sequence

$$\mathbf{vec} : \mathbb{R}^{l\times m} \rightarrow \mathbb{R}^{lm\times 1}, \tag{1}$$

where l, m denote any shape. For an arbitrary $H = [h_{ij}] \in \mathbb{R}^{l\times m}$, $\mathbf{vec}(\cdot)$ yields its 1-d patched format, as

$$\mathbf{vec}(H) = \begin{bmatrix} h_{11} \\ \vdots \\ h_{l1} \\ \vdots \\ h_{1m} \\ \vdots \\ h_{lm} \end{bmatrix} \tag{2}$$

In our setting, h_{ij} represents the embedding derived from an image patch.

Second, we define a commutation matrix $C_{lm} \in \mathbb{R}^{lm\times lm}$, which fulfills

$$C_{lm}\mathbf{vec}(H) = \mathbf{vec}\left(H^T\right). \tag{3}$$

We can easily validate that $C_{lm}^T = C_{lm}^{-1} = C_{lm}$ [7]. Therefore, C_{lm} is an orthogonal matrix.

Finally, according to [6], we have that for matrices $A_{n\times p}$, $B_{p\times q}$, and $C_{q\times m}$, the theorem holds that

$$\mathbf{vec}(ABC) = \left(C^{\mathrm{T}} \otimes A\right)\mathbf{vec}(B), \tag{4}$$

where $\otimes$ is Kronecker product.

Spatial Transformation Operation Given an input image $I \in \mathbb{R}^{H\times W}$ (channel dimension omitted for simplicity), we consider a spatial transformation operation (i.e., flipping or rotation) as a mapping of each individual pixel $(i, j) \in I$. Specifically, for flipping, we have three cases

$$\begin{aligned} \text{horizontal flip: } (i,j) &\rightarrow (i, W-j), \\ \text{vertical flip: } (i,j) &\rightarrow (H-i, j), \\ \text{horizontal \& vertical flip: } (i,j) &\rightarrow (H-i, W-j), \end{aligned}$$

which can be represented by permutation operations.

Likewise, for rotation, we have

$$\begin{aligned} 90^\circ\text{rotate: } (i,j) &\rightarrow (W-j, H-i), \\ 180^\circ\text{rotate: } (i,j) &\rightarrow (j, H-i), \\ 270^\circ\text{rotate: } (i,j) &\rightarrow (W-j, I), \end{aligned}$$

while each case can be further considered as a matrix transpose followed by a flipping operation.

As such, we unify the above operations into matrix transformations. Given the feature map $X \in \mathbb{R}^{h\times w}$ of the said image I obtained from e.g., ViT, where $h \times w = n$ (n patches inside the transformer), we have

$$\text{flip: } X \rightarrow P_h X P_w, \tag{5}$$

$$\text{rotation: } X \rightarrow P_h X^T P_w, \tag{6}$$

where $P_h \in \mathbb{R}^{h\times h}$ and $P_w \in \mathbb{R}^{w\times w}$ are permutation matrices in the x and y directions respectively.

Self-attention Matrices Here, we ask the question — how will the self-attention matrices in the transformer change according to a spatial transformation operation on the image?

We assume Q^s, K^s as the two projected feature maps of the input image, but in the 2-d shapes before flattening. Per the transformer attention design [10], we denote $Q^s = XW_Q$ and $K^s = XW_K$, where Q^s, K^s are of dimension $\mathbb{R}^{h\times w\times d}$ with d being the feature dimension. Here, W_Q and W_K project the embedded input image into two latent spaces, then we use $\mathbf{vec}(\cdot)$ to flatten Q^s and K^s. The self-attention matrix of the original image before transformation is then defined as

$$A = \mathbf{vec}(Q^s)(\mathbf{vec}(K^s))^T \in \mathbb{R}^{n\times n}, \tag{7}$$

where we omit the class token for simplicity. Then, we write out the matrix after the transformation. For flipping, the augmented self-attention matrix is formulated as

$$A' = (\mathbf{vec}(P_h Q^s P_w))(\mathbf{vec}(P_h K^s P_w))^T \in \mathbb{R}^{n\times n}, \tag{8}$$

which, per Equation 4, can be further derived as

$$\begin{aligned} A' &= (\mathbf{vec}(P_h Q^s P_w))(\mathbf{vec}(P_h K^s P_w))^T \\ &= (P_w^T \otimes P_h)\mathbf{vec}(Q^s)((P_w^T \otimes P_h)\mathbf{vec}(K^s))^T \\ &= (P_w^T \otimes P_h)\mathbf{vec}(Q^s)\mathbf{vec}(K^s)^T(P_w^T \otimes P_h)^T \\ &= (P_w^T \otimes P_h)A(P_w^T \otimes P_h)^T. \end{aligned} \tag{9}$$

For rotation, the augmented self-attention is formulated as

$$A' = (\mathbf{vec}(P_h(Q^s)^T P_w))(\mathbf{vec}(P_h(K^s)^T P_w))^T \in \mathbb{R}^{n\times n}. \tag{10}$$

Following the axiom of 3 where $C \in \mathbb{R}^{n\times n}$ is an commutation matrix, Equation 10 can be rewritten as

$$\begin{aligned} A' &= (\mathbf{vec}(P_h(Q^s)^T P_w))(\mathbf{vec}(P_h(K^s)^T P_w))^T \\ &= (P_w^T \otimes P_h)C\mathbf{vec}(Q^s)((P_w^T \otimes P_h)C\mathbf{vec}(K^s))^T \\ &= (P_w^T \otimes P_h)C\mathbf{vec}(Q^s)\mathbf{vec}(K^s)^T C^T(P_w^T \otimes P_h)^T \\ &= (P_w^T \otimes P_h)CAC^T(P_w^T \otimes P_h)^T. \end{aligned} \tag{11}$$

Table 1. Ablation of different image augmentation methods. We report mIoU of seeds on PASCAL VOC *train* set.

Augmentation	mIoU
Baseline (no aug)	57.7
Resize	59.2
Rotation	61.1
Horizontal flip + resize	61.6
Horizontal flip + vertical flip	63.6
Horizontal flip + patch hiding	65.8
Horizontal flip + gray scale	63.8
Horizontal flip	67.3

Table 2. Ablation of different distance metrics for regularization loss. We report mIoU of seeds on PASCAL VOC *train* set.

Loss	mIoU
L2	62.5
Smooth L1	62.5
L1	67.3

Table 3. Computational comparison. The training memory and test FPS are tested on an RTX2080Ti GPU with a batch size of 1.

Method	Backbone	Resolution	Train memory(MB)	Test FPS
PSA [2]	ResNet38	384	3082	0.98
MCTformer [12]	Deit-S	224	1500	7.10
Ours	Deit-S	224	1580	3.61
Ours	Vitb-hybrid-B	384	5260	2.33

Transformation Inversion At last, we obtain the formulation to invert the transformation on the attention matrices. Following Equation 9 and Equation 11, the transformation inversion is in a unified form

$$f^{-1}(A') = C^T(P_w \otimes P_h^T)A'(P_w \otimes P_h^T)^T C, \qquad (12)$$

where f^{-1} is the inversion transformation. $C \in \mathbb{R}^{n\times n}$ is a commutation matrix for rotation and an identity matrix when flipping. Note that such a formulation enables inversion of a wide range of possible image transformations that can be described with permutation matrices, though few may be helpful augmentations. To this end, $f^{-1}(A')$ and A are spatially equivalent and we can directly calculate the distance between the two attention matrices to apply ACR.

3. Analysis of different image augmentation

Several image augmentations are adopted in ACR to transform the second view, we report the performance of them in Table 1. As shown, we ablate several combinations of image augmentations. It is observed that all image augmentations achieve performance improvements over the baseline, which validates the effectiveness of ACR. Second, we found that horizontal flip on its own achieves the best result (67.3% mIoU). In future work, we will further investigate how different augmentations affect the performances.

4. More qualitative results

We show more qualitative results of the class localization maps provided by ACR. In Fig. 2, we show class localization maps of images with simple scenes. In Fig. 3, we show that ACR also generates high-quality class localization maps with multiple classes. In the bottom row of Fig. 3, we show a failure case of an image containing a horse and a rider. Competitive relationships between the class activation are not investigated in this paper, so when we have multiple connected objects that have similar appearances or belong to the co-occurring classes, the affinity refinement may lead to over-activation. We will investigate this issue in future work.

Pair-wise relationships, or affinity, between image regions are inherently encoded in the attention matrix of the vision transformer. The model is encouraged to capture consistent pair-wise affinity by our region affinity regularization. We display the class localization maps and learned affinity matrices. in Fig. 4. The baseline model with classification loss only produces noisy localization maps, which is consistent with other methods, such as [12, 9, 8], Further, if we only apply activation consistency regularization such as (ACT regu), model can correctly localize targeted objects but fails to capture precise object shapes. As shown, some particular tokens are still causing the affinity matrices to become disorganized, which indicates that simple activation consistency such SEAM [11] is not adequate and further affinity consistency is necessary. Finally, our ACR can generate high-quality object localization maps as the affinity regularization plays a key role in ensuring consistent appearance of object features, which in turn enhances segmentation performance In addition, We show qualitative examples of the learned affinity in Fig. 5. We select three positions of the image which are marked as red crosses and show their related affinity. As shown, the learned affinity highly corresponds to semantic entities and shows accurate boundaries. For example, the background (wall, sky, ground) and foreground objects are clearly separated. Such results indicate that our vision-transformer-based ACR learns high-quality affinity and can effectively refine the class localization maps by propagating related pixels.

Finally, we show qualitative examples of segmentation predictions in Fig. 6.

5. Analysis of Regularization Loss

Given two spatially equal attention matrices, we measure the distance between them. In Table 2, we ablate different types of distance evaluation methods and report the mIoU of the class localization maps on the PASCAL VOC *train* set. As shown, we empirically found that L1 distance achieves the best result.

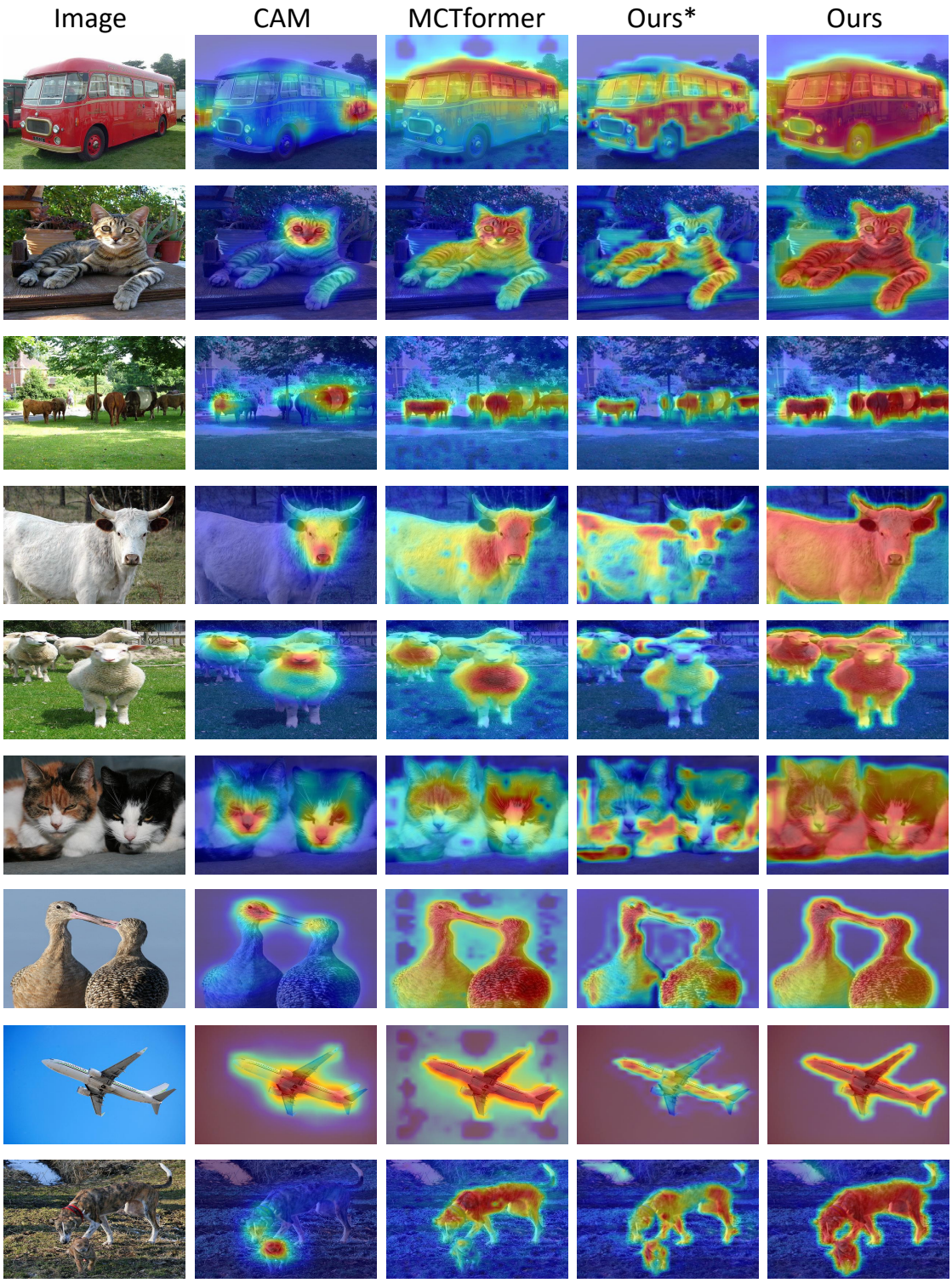

Figure 2. Qualitative examples of class localization maps of ACR. CAM: [14]. MCTformer: [12]. Ours*: our maps without affinity refinement. Ours: our final class localization maps.

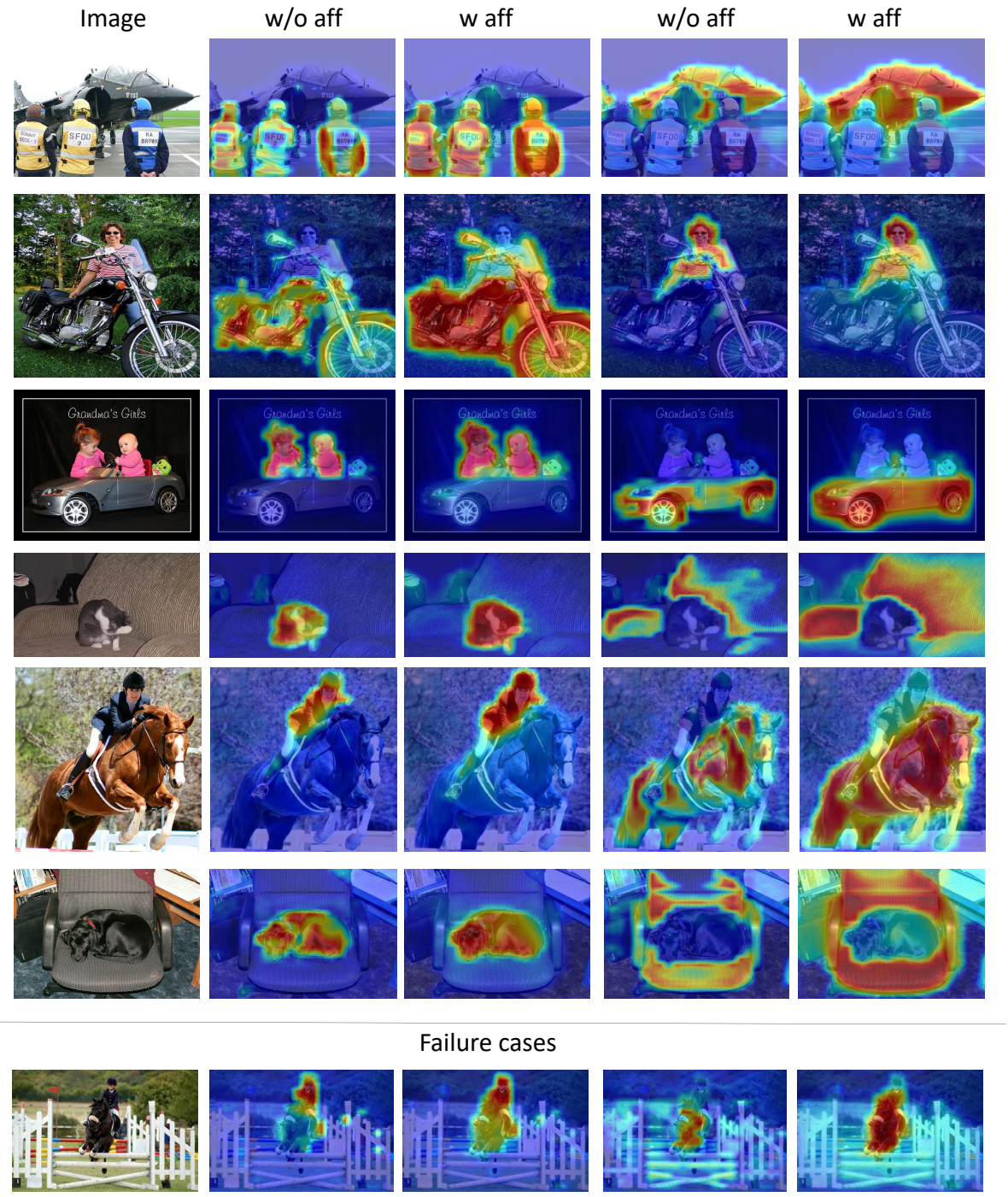

Figure 3. Qualitative examples of our class localization maps with multiple classes. We show the results without and with affinity refinement. In the bottom, we present a failure case.

6. Analysis of Efficiency

In Table. 3, we compare our method with CNN-based PSA [2] and Transformer based MCTformer [12]. During

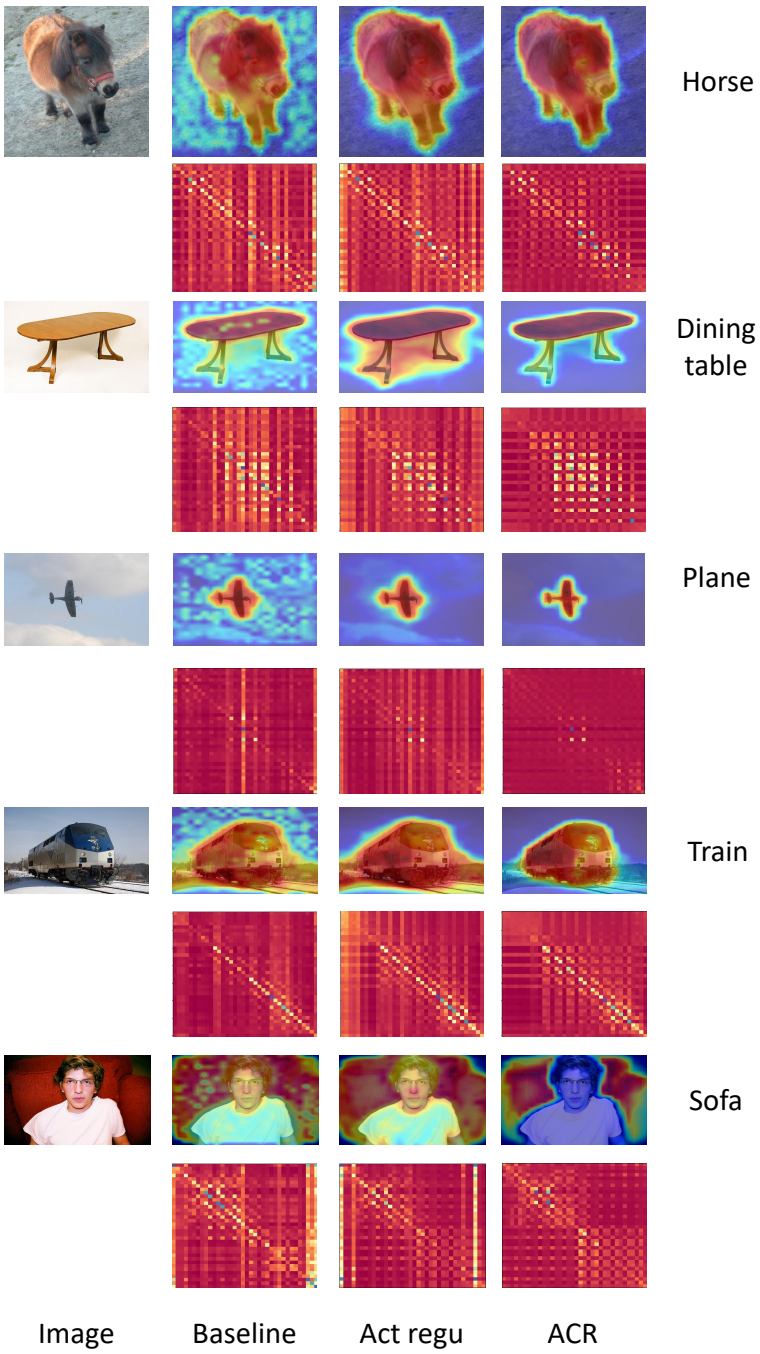

Figure 4. Qualitative examples of the class localization maps and the learned affinity matrices. Baseline: the model is trained with only classification. Act regu: the model is trained with only activation consistency regularization. ACR: the model is trained with our ACR that contains both consistency regularization. The baseline model with classification loss only generates noisy localization. With our activation regularization (Act regu), the model can correctly localize targeted objects but fails to capture precise boundaries. Finally, our ACR can generate high-quality object localization maps, showing clearly the performance increase that arises from affinity consistency regularization.. The affinity matrices are down-sampled for readability.

training, our memory usages are similar under the same network and image size. Our inference speed is significantly faster than that of PSA. However, our inference speed is slower than MCTformer due to the fact that MCT-

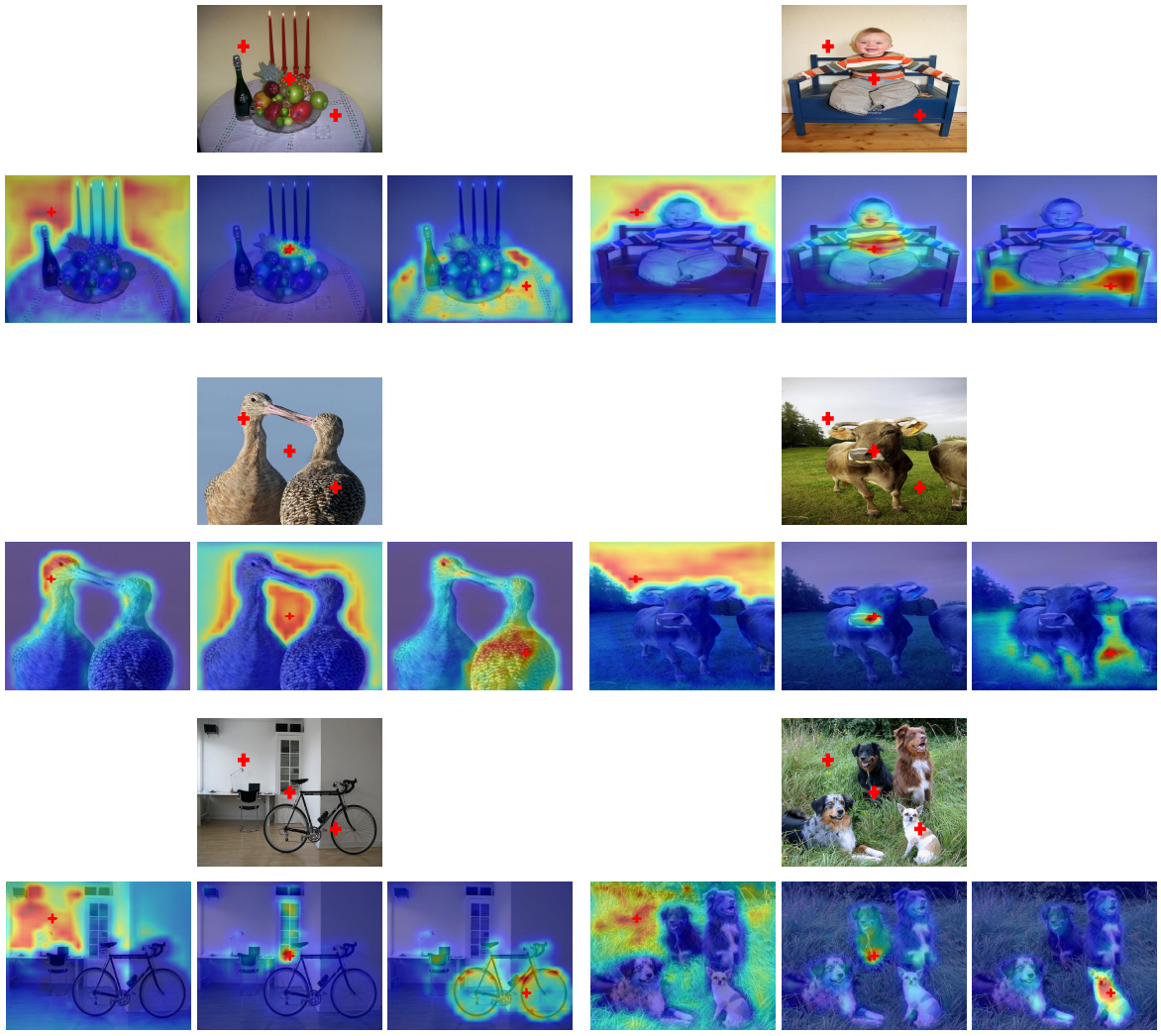

Figure 5. Qualitative examples of the learned affinity of ACR. Three source pixels are marked as red crosses. The region affinity related to these three source pixels is demonstrated respectively. Each source pixel is highly correlated with its semantically matched regions.

former adopts multiple class tokens, whereas we maintain the model structure with a single class token and calculate gradients to generate class-wise localization.

7. Limitations and Future Research

We discuss the limitations and future research possibilities of our method in this section. First, competitive relationships between class activation are not investigated in this paper, instead, our regularization is directly applied to class-indifferent attention matrices. Thus, affinity refinement may lead to over-activation when multiple connected objects share similar appearances or belong to co-occurring classes. In future work, we will investigate how to connect the self-attention mechanism with the semantic relations between the classes so we can generate more class discriminative localization maps. Second, as discussed in the main paper, our class localization maps are generally over-activated as the FP is consistently higher than the FN. It indicates that the incompleteness issue is effectively mitigated by ACR. However, current pseudo generation methods [2, 1]are designed for under-activated seeds, i.e., they require the seeds to have a high precision rather than recall. It might be the reason that our pseudo label improvement is not as significant as our class localization maps. A compatible solution for over-activation is expected in the future and it would potentially improve the segmentation results of ACR even further.

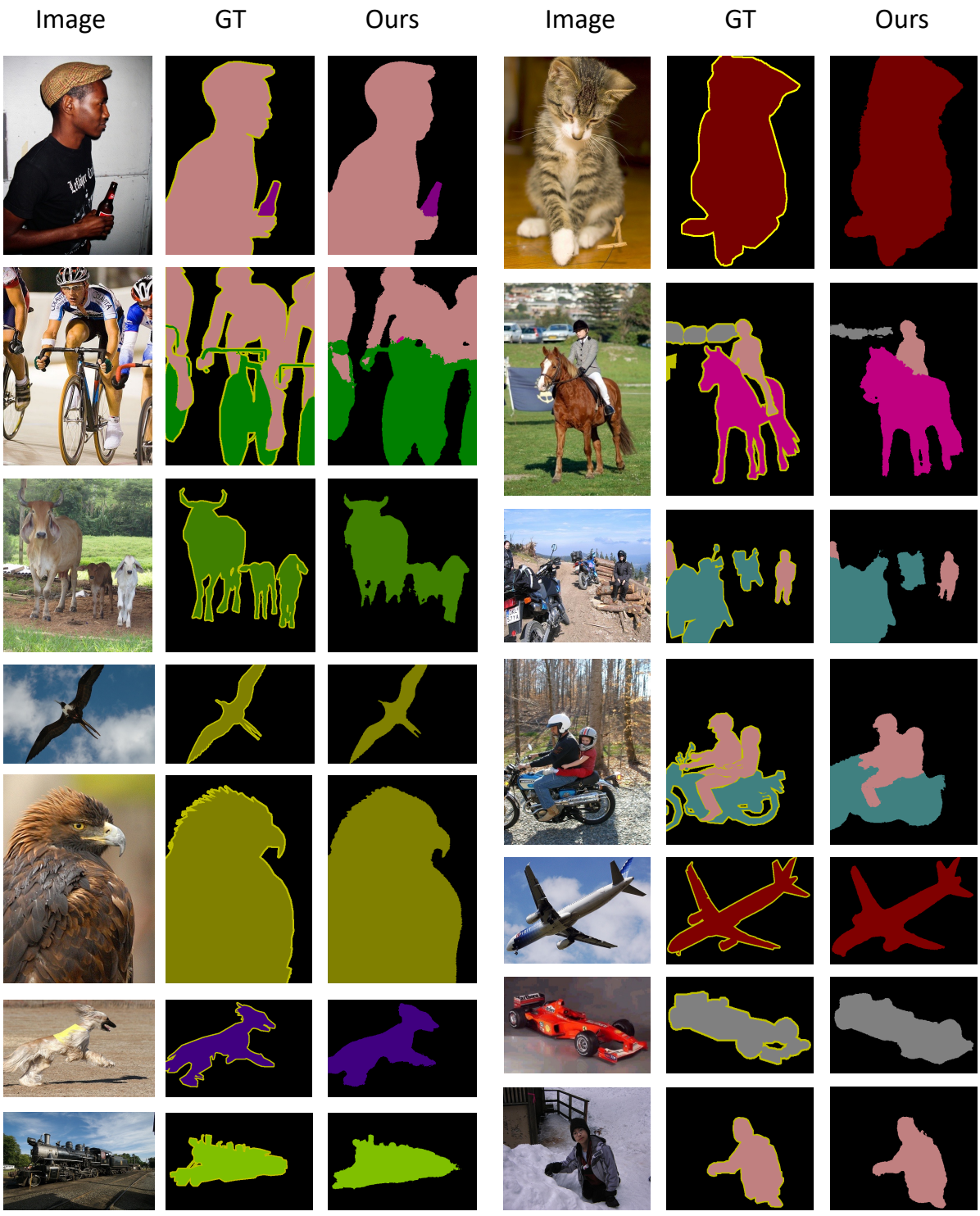

Figure 6. Qualitative examples of the segmentation predictions

8. Per class results of PASCAL VOC and MS COCO

We report the per class IoU of PASCAL VOC *val* set and MS COCO *val* set in Table 4 and Table 5.

References

[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2209–2218, Long Beach, CA, USA, 2019. Computer Vision Foundation / IEEE. 7

[2] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4981–4990. Computer Vision Foundation / IEEE, 2018. 3, 5, 7

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[4] Hyeokjun Kweon, Sung-Hoon Yoon, Hyeonseong Kim, Daehee Park, and Kuk-Jin Yoon. Unlocking the potential of ordinary classifier: Class-specific adversarial erasing framework for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6994–7003, October 2021. 10

[5] Jungbeom Lee, Jooyoung Choi, Jisoo Mok, and Sungroh Yoon. Reducing information bottleneck for weakly supervised semantic segmentation. *Advances in Neural Information Processing Systems*, 34, 2021. 10

[6] Jan R Magnus and Heinz Neudecker. The commutation matrix: some properties and applications. *The Annals of Statistics*, 7(2):381–394, 1979. 2

[7] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008. 2

[8] Lixiang Ru, Yibing Zhan, Baosheng Yu, and Bo Du. Learning affinity from attention: End-to-end weakly-supervised semantic segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16846–16855, 2022. 3

[9] Weixuan Sun, Jing Zhang, Zheyuan Liu, Yiran Zhong, and Nick Barnes. Getam: Gradient-weighted element-wise transformer attention map for weakly-supervised semantic segmentation. *arXiv preprint arXiv:2112.02841*, 2021. 3

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[11] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, June 2018. 3

[12] Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, and Dan Xu. Multi-class token transformer for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4310–4319, 2022. 3, 4, 5, 10

[13] Fei Zhang, Chaochen Gu, Chenyue Zhang, and Yuchao Dai. Complementary patch for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7242–7251, October 2021. 10

[14] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929. Computer Vision Foundation / IEEE, 2016. 4

Table 4. Per-class results on PASCAL VOC *val* set.

Class	bkg	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mIoU
CPN [13]	89.9	75.1	32.9	**87.8**	**60.9**	69.5	**87.7**	79.5	**89.0**	28.0	80.9	34.8	83.4	79.7	74.7	66.9	56.5	82.7	44.9	**73.1**	45.7	67.8
Kweon et al. [4]	90.2	82.9	35.1	86.8	59.4	70.6	82.5	78.1	87.4	30.1	79.4	45.9	83.1	83.4	**75.7**	73.4	48.1	**89.3**	42.7	60.4	52.3	68.4
Ours	**91.5**	**85.2**	**39.7**	85.8	60.4	**77.0**	87.4	**80.1**	87.9	**30.3**	**84.2**	**50.7**	**83.5**	**85.8**	74.1	**73.5**	**59.7**	83.8	**45.1**	72.5	**55.5**	**71.2**

Table 5. Per-class results on MS COCO *val* set.

Class	MCTformer [12]	RIB [5]	ours	Class	MCTformer [12]	RIB [5]	ours
background	82.4	82.0	**82.7**	wine glass	27.0	27.5	**48.2**
person	**62.6**	56.1	47.0	cup	29.0	27.4	**42.6**
bicycle	47.4	**52.1**	50.4	fork	13.9	**15.9**	12.6
car	**47.2**	43.6	44.6	knife	12.0	14.3	**16.1**
motorcycle	63.7	67.6	**68.4**	spoon	6.6	8.2	**9.5**
airplane	64.7	61.3	**70.2**	bowl	22.4	20.7	**26.5**
bus	64.5	68.5	**71.1**	banana	63.2	59.8	**64.3**
train	**64.5**	51.3	56.4	apple	44.4	**48.5**	**48.5**
truck	**44.8**	38.1	37.6	sandwich	39.7	36.9	**51.0**
boat	**42.3**	**42.3**	37.1	orange	63.0	62.5	**63.1**
traffic light	**49.9**	47.8	37.4	broccoli	51.2	45.4	**53.8**
fire hydrant	73.2	73.4	**74.9**	carrot	40.0	34.6	**44.3**
stop sign	**76.6**	76.3	65.2	hot dog	**53.0**	49.7	52.1
parking meter	64.4	**68.3**	50.8	pizza	62.2	58.9	**79.3**
bench	32.8	39.7	**43.1**	donut	55.7	53.1	**65.5**
bird	**62.6**	57.5	60.2	cake	47.9	40.7	**52.6**
cat	78.2	72.4	**78.4**	chair	**22.8**	20.6	18.7
dog	68.2	63.5	**72.0**	couch	35.0	36.8	**39.9**
horse	65.8	63.6	**67.5**	potted plant	13.5	17.0	**22.5**
sheep	70.1	69.1	**70.4**	bed	48.6	46.2	**51.0**
cow	68.3	68.3	**71.4**	dining table	12.9	11.6	**19.6**
elephant	**81.6**	79.5	81.2	toilet	63.1	63.9	**65.7**
bear	80.1	76.7	**82.7**	tv	47.9	39.7	**50.7**
zebra	**83.0**	80.2	82.1	laptop	49.5	48.2	**54.6**
giraffe	**76.9**	74.1	76.2	mouse	13.4	**22.4**	11.8
backpack	14.6	**18.1**	13.3	remote	**41.9**	38.0	37.4
umbrella	61.7	60.1	**64.4**	keyboard	49.8	50.9	**53.5**
handbag	4.5	**8.6**	8.2	cellphone	**54.1**	**54.1**	53.2
tie	25.2	**28.6**	27.1	microwave	38.0	45.2	**46.7**
suitcase	46.8	**49.2**	48.3	oven	29.9	**35.9**	32.7
frisbee	43.8	53.6	**57.0**	toaster	0.0	**17.8**	0.0
skis	12.8	9.7	**14.1**	sink	28.0	**33.0**	30.4
snowboard	**31.4**	29.4	23.7	refrigerator	40.1	**46.0**	32.9
sports ball	9.2	**38.0**	21.5	book	32.2	31.1	**33.2**
kite	26.3	37.1	**47.1**	clock	43.2	41.9	**52.6**
baseball bat	0.9	**15.3**	11.0	vase	22.6	27.5	**31.4**
baseball glove	0.7	**8.1**	7.1	scissors	32.9	41.0	**42.4**
skateboard	7.8	**31.8**	26.0	teddy bear	61.9	**62.0**	60.3
surfboard	**46.5**	29.2	38.6	hair drier	0.0	**16.7**	0.0
tennis racket	1.4	**48.9**	21.0	toothbrush	11.1	21.0	**31.0**
bottle	31.1	33.1	**38.5**	mIoU	42.0	43.8	**45.0**