Supplementary Material for: F-CAM: Full Resolution Class Activation Maps via Guided Parametric Upscaling

Soufiane Belharbi[1], Aydin Sarraf[3], Marco Pedersoli[1], Ismail Ben Ayed[1], Luke McCaffrey[2], and Eric Granger[1]
[1] LIVIA, Dept. of Systems Engineering, École de technologie supérieure, Montreal, Canada
[2] Goodman Cancer Research Centre, Dept. of Oncology, McGill University, Montreal, Canada
[3] Ericsson, Global AI Accelerator, Montreal, Canada
soufiane.belharbi.1@ens.etsmtl.ca, luke.mccaffrey@mcgill.ca, aydin.sarraf@ericsson.com,
{marco.pedersoli, ismail.benayed, eric.granger}@etsmtl.ca

We provide in this supplementary document:
1) General description of CRF loss $\mathcal{R}$ [15].
2) Failure cases (Sec.2).
3) Ablation study (Sec.3).
4) Simulation of the impact of downscale factor of CAMs over the localization performance (Sec.4).
5) Runtime inference for WSOL baselines and our method (Sec.5) and number of parameters per model.
6) `MaxBoxAccV2` performance over CUB (Tab.3).
7) Classification performance of different WSOL baselines (Tab.5).
8) Convergence curves of WSOL baseline training. (Fig.9).
9) Comparison of localization performance curves (Fig.7,8).
10) Comparison to CAMs activations distributions (Fig.5, 6).

1. General description of CRF loss

Given an input image $\boldsymbol{X}$ and the softmax activation $\boldsymbol{S}$ of the decoder, the CRF loss is formulated [15] as,

$$\mathcal{R}(\boldsymbol{S}, \boldsymbol{X}) = \sum_{t=1}^{t=2} {\boldsymbol{S}^t}^\top \, \boldsymbol{W} \, (\boldsymbol{1} - \boldsymbol{S}^t) \;, \tag{1}$$

where $\boldsymbol{W}$ is an affinity matrix where $\boldsymbol{W}[i, j]$ captures the color similarity and proximity between pixels i, j in the image $\boldsymbol{X}$. We consider using Gaussian kernel to capture color and spatial similarities [8]. We use the permutohedral lattice [1] for fast computation of $\boldsymbol{W}$. Minimizing Eq.1 pushes the decoder to produce consistent activations for nearby pixels with similar color.

2. Failure cases.

Fig.1 illustrates few examples over OpenImages test set where our method failed to localize the correct object. The fine-tuning of our method is guided mainly by the activations of the WSOL baseline. When the activations largely miss the correct object, our method learns on wrong supervisory signal leading to false localization. Detecting these cases and dealing with them remains an open issue in this work. This scenario goes under learning from highly noisy supervisory signals which is still a growing field [14].

3. Ablation study

Tab.1 shows the impact of adding different terms in the pixel alignment loss in our training loss on all backbones and on both datasets using CAM* method. We observe that simply adding sampling regions of foreground and background yields a large improvement of the baseline. This shows the importance of using supervisory signals at pixel level to guide CAMs. Baseline methods use only classification signal which is a global information. Adding local guidance helps better discerning foreground from background. Adding a CRF term and size constraint helps better extending the foreground while respecting object's boundaries yielding more improvement. We report in Fig.2 the impact of the hyper-parameter n^- on the localization performance. Note that n^- describes the extent of the background region we are allowed to sample from. It is interconnected with the object size. Datasets with large objects result in small background and small objects lead to large background. This is reflected in these curves. On CUB, often objects are small. Therefore, assuming large background is safe. As a result, large values of n^- yield better results. On the other hand, objects in OpenImages dataset

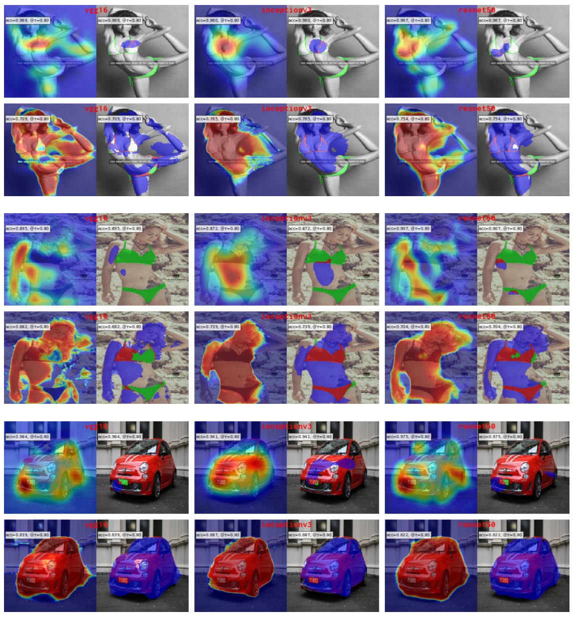


Figure 1: Failed cases of our method combined with CAM* over OpenImages test set. Colors: green false negative, red : true positive, blue: false positive. $\tau = 0.8$.

tend to be large leaving small background, making small values of n^- more efficient. This gives us an intuition for better range for n^-.

	CUB (`MaxBoxAcc`)				OpenImages (`PxAP`)			
Methods	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean
CAM* [20]	61.6	58.8	71.5	63.9	53.0	62.7	56.8	57.5
CAM* [20] + SR	84.2	73.0	82.2	79.8	64.5	64.1	63.8	64.1
CAM* [20] + SR + ASC	82.9	74.1	83.2	80.0	63.9	63.4	62.0	63.1
CAM* [20] + SR + CRF	84.6	78.9	86.1	83.2	66.3	68.3	67.5	67.3
CAM* [20] + SR + CRF + ASC	87.3	82.0	90.3	86.5	67.8	71.9	72.1	70.6
Improvement	+25.7	+23.2	+18.8	+22.5	+14.8	+9.2	+15.3	+12.8

Table 1: Ablation study of different terms in the pixel alignment loss over CAM* baseline. The bottom line in green is the improvement over the WSOL baseline CAM* (top green line), when combined with our full method 4th green line.

4. Simulation of the impact of downscale factor of CAM over `PxAP` performance

The size of the CAM is influenced by many operations in the model including convolution, and pooling operations. Standard models such as Resnet family [5] downscales the CAM by a factor z up to $z = 32$. In order to assess the impact of such downscale factor over the localization performance obtained through a CAM, we propose a simple simulation. For this simulation, we consider the dataset OpenImages as it provides the pixel-wise annotation in order to evaluate the `PxAP` performance. Instead of working directly on a model, we substitute the model's predicted low resolution CAM by a downsampled, with factor z, version of the true mask. Then, we measure the `PxAP` performance between the upscaled CAM and the true mask. This yields an almost perfect low resolution CAM but we assume it was predicted by the model. The procedure is illustrated in Fig.4. The results are presented in Fig.3. This figure shows that the low resolution size of CAMs is a major bottleneck in localization. Even when the low resolution CAM is directly the downscaled version of the true mask, upscaling back to the high resolution does not yield the exact mask due to loss of information when downscaling. `PxAP` values in Fig.3 could be seen as an upper bound for pixel-wise localization in function of the scale factor. This implies that our method with full resolution can still yield better performance. These results suggest as well that for a better localization performance, it is better to use a low scale factor.

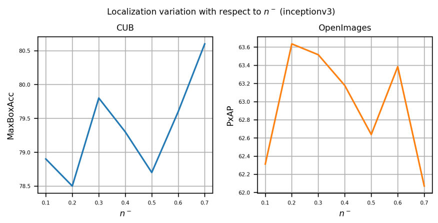


Figure 2: Impact of n^- on the localization performance over CUB and OpenImages validation with CAM* baseline + ours. Random runs are done while the rest of hyper-parameters are fixed.

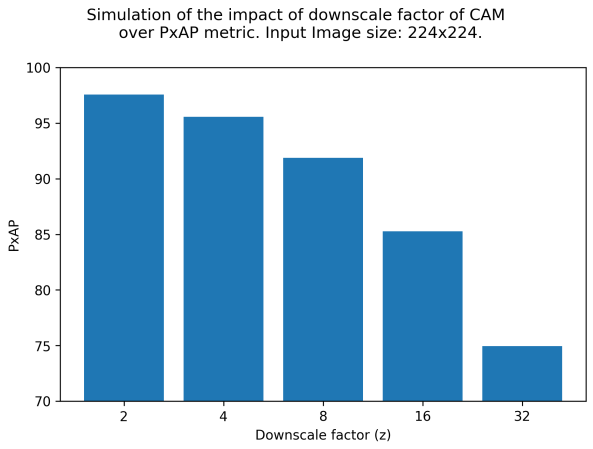


Figure 3: The localization performance over the test set of OpenImages dataset using `PxAP` metric with respect to the downscale factor z of the simulated CAM. The simulation process is presented in Fig.4.

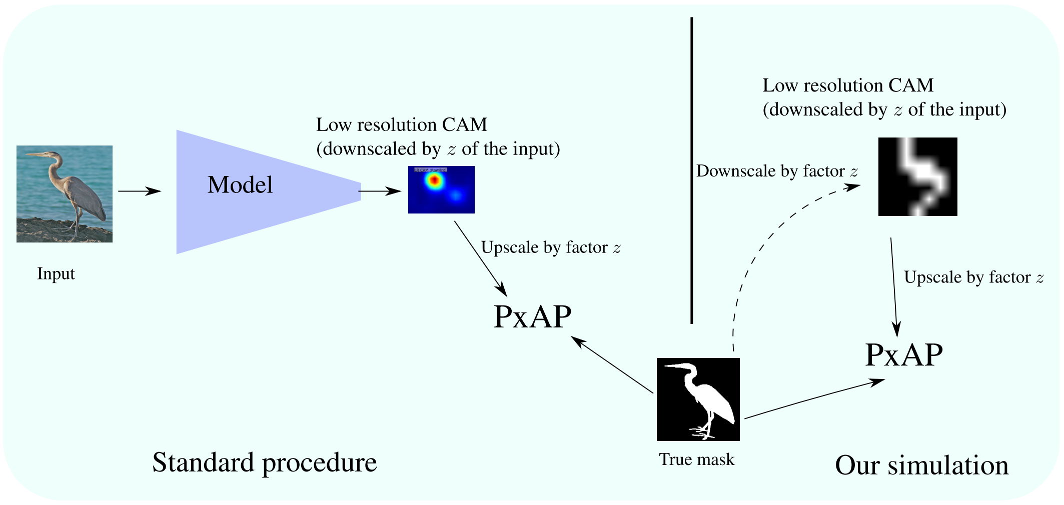


Figure 4: Simulation to evaluate the impact of CAM size over the localization performance using `PxAP` metric. In our simulation, we substitute the low resolution CAM predicted by the model by downscaled version of the true mask by a factor z allowing us to quickly change the scale and assess its impact on the localization performance using `PxAP` metric. The scaling procedure is performed using bilinear interpolation.

5. Runtime analysis

We report in Tab.4 the time required to build a CAM for all the studied WSOL finetuned baselines and other methods. While adding a decoder to a standard classifier increases the number of parameters, the inference time is still better than average finetuned baselines and competitive with other CAM methods. Because the inference is achieved through a single forward with fully convolutional operations, our method is expected to be fast. Other methods may require a forward and a backward to estimate a CAM. Our method is still competitive to ACoL, SPG, and ADL methods. We included other methods ScoreCAM [16], SS-CAM [19], and IS-CAM [9]. Their slow runtime prevented us from considering them as baselines.

Backbone	VGG	Inception	ResNet
Encoder			
Layer 1	128	64	64
Layer 2	256	80	256
Layer 3	1024	288	512
Layer 4	–	768	1024
Layer 5	–	1024	2048
Decoder			
Layer 1	256	256	256
Layer 2	128	128	128
Layer 3	64	64	64
Layer 4	–	32	32
Layer 5	–	16	16

Table 2: Architecture details. We use the same common backbones VGG16, InceptionV3, and Resnet50 without modification. Each backbone has its own definition of a layer and specific number of layers.

	CUB (`MaxBoxAccV2`)			
Methods	VGG	Inception	ResNet	Mean
CAM [20]*(cvpr,2016)*	63.7	56.7	63.0	61.1
HaS [13]*(iccv,2017)*	63.7	53.4	64.7	60.6
ACoL [18] *(cvpr,2018)*	57.4	56.2	66.5	60.0
SPG [19]*(eccv,2018)*	56.3	55.9	60.4	57.5
ADL [3] *(cvpr,2019)*	66.3	58.8	58.4	61.1
CutMix [17]*(eccv,2019)*	62.3	57.5	62.8	60.8
Best WSOL	66.3	58.8	66.5	61.1
FSL baseline	71.6	86.6	82.4	80.2
Center baseline	59.7	59.7	59.7	59.7
ICL [7]*(accv,2020)*	66.7	60.3	63.2	63.4
CAM* [20] *(cvpr,2016)*	57.0	54.4	62.1	57.8
GradCAM [12] *(iccv,2017)*	62.7	57.1	63.3	61.0
GradCAM++ [2] *(wacv,2018)*	73.8	60.7	70.2	68.2
Smooth-GradCAM++ [11] *(corr,2019)*	64.1	59.7	66.6	63.4
XGradCAM [4] *(bmvc,2020)*	62.8	56.7	63.2	60.9
LayerCAM [6] *(ieee,2021)*	74.1	62.6	72.6	69.7
CAM* [20] + ours	79.1	71.2	79.4	76.5
GradCAM [12] + ours	79.5	76.2	80.8	78.5
GradCAM++ [2] + ours	84.1	73.1	82.7	79.9
Smooth-GradCAM++ [11] + ours	83.1	74.0	81.6	79.5
XGradCAM [4] + ours	80.1	70.6	80.0	76.9
LayerCAM [6] + ours	84.3	73.9	82.7	80.3
Best WSOL + ours	84.3	76.2	82.7	80.3

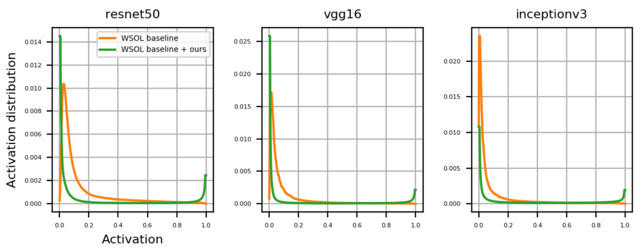
Table 3: Performance on CUB using `MaxBoxAccV2` metric.

Backbones (encoders)	VGG16				Inception				ResNet50			
Methods	#PCL	#NFM	SFM	#PDEC	#PCL	#NFM	SFM	#PDEC	#PCL	#NFM	SFM	#PDEC
Details	≈19.6	1024	28x28	≈23.1	≈25.6	1024	28x28	≈5.7	≈23.9	2048	28x28	≈9
CAM* [20]	.2ms				.2ms				.3ms			
GradCAM [12]	7.7ms				21.1ms				27.8ms			
GradCAM++ [2]	23.5ms				23.7ms				28.0ms			
Smooth-GradCAM [11]	62.0ms				150.7ms				136.2ms			
XGradCAM [4]	2.9ms				19.2ms				14.2ms			
LayerCAM [6]	3.2ms				18.2ms				17.9ms			
Mean	16.6ms				38.8ms				37.4ms			
ours + STDCL	6.2ms				25.5ms				18.5ms			
ACoL [18]	12.0ms				19.2ms				24.9ms			
SPG [19]	11.0ms				18ms				23.9ms			
ADL [3]	6.4ms				16.0				14.4ms			
ScoreCAM [16]	1.9sec				3.4sec				9.3sec			
SSCAM [10]	1min45sec				2min16sec				5min49sec			
IS-CAM [9]	30.1sec				39.0sec				1min39sec			

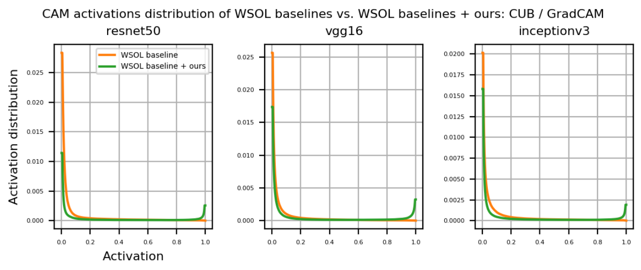
Table 4: Time required to build CAMs of different WSOL methods. **STDCL**: standard classifier = encoder (VGG16, Inception, ResNet50) + global average pooling. **#PCL** (millions): number of the parameters of the classifier. **#NFM**: number of the feature maps at the top layer. **SFM**: size of the feature maps at the top layer. **#PDEC** (millions): number of the parameters of the decoder. **Time**: time necessary top build a full size CAM over an idle Tesla P100 GPU for one random RGB image of size 224×224 with 200 classes. Methods SSCAM [10] ($N = 35, \sigma = 2$), IS-CAM [9] ($N = 10$), IS-CAM [9] ($N = 10$) are evaluated with batch size 32 with their original hyper-parameters ($N,$ and σ).

	CUB				OpenImages			
Methods	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean
CAM [20]	26.8	61.8	58.4	49.0	67.3	36.6	72.6	58.8
HaS [13]	70.9	69.9	74.5	71.8	60.0	68.4	74.0	67.5
ACoL [18]	56.1	71.6	64.0	63.9	68.2	40.7	70.7	59.9
SPG [19]	63.1	58.8	37.8	53.2	71.7	43.5	65.4	60.2
ADL [3]	31.1	45.5	32.7	36.4	66.1	46.6	56.1	56.3
CutMix [17]	29.2	70.2	55.9	51.8	68.1	53.1	73.7	65.0
CAM* [20]	49.3	65.5	65.1	59.9	69.1	61.2	73.2	67.8
GradCAM	24.8	65.4	42.2	44.1	69.0	54.0	72.4	65.1
GradCAM++	24.8	65.2	42.2	44.0	70.3	69.6	72.3	70.7
Smooth-GradCAM	24.8	65.3	42.2	44.1	69.0	54.0	67.0	63.3
XGradCAM	24.8	65.4	65.1	51.7	69.0	69.4	72.0	70.1
LayerCAM	24.8	65.0	51.2	47.0	70.3	53.9	72.2	65.4

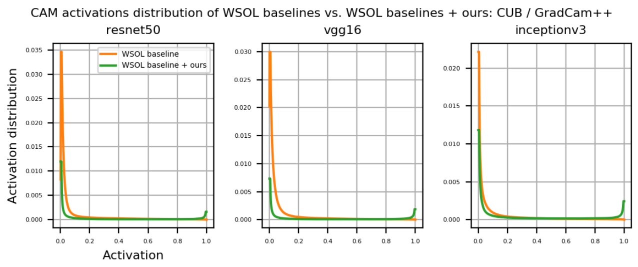
Table 5: Classification performance of WSOL baseline methods. Model selection is performed over localization performance `MaxBoxAcc` and `PxAP`.



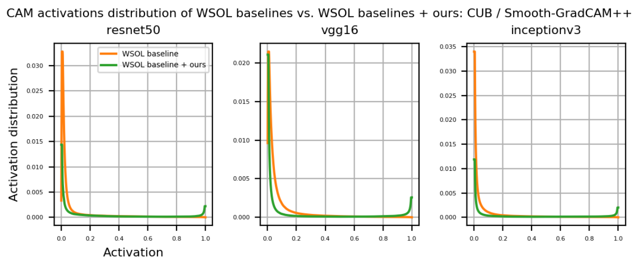
(a) CAM*



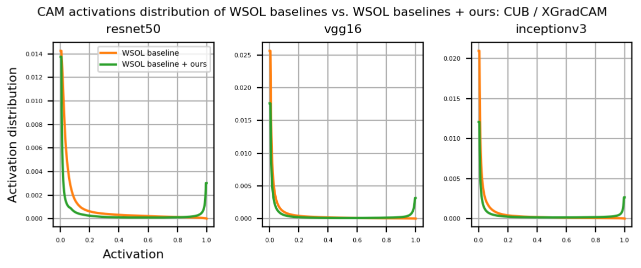
(b) GradCAM



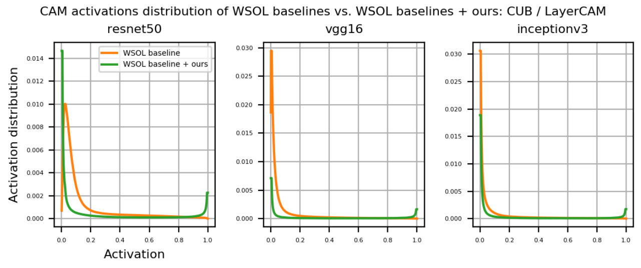
(c) GradCAM++



(d) Smooth-GradCAM++

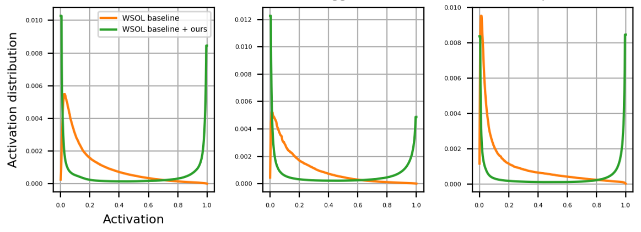


(e) XGradCAM

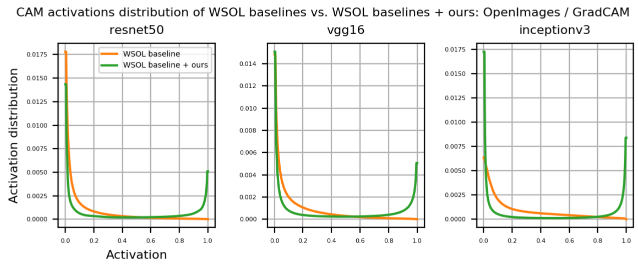


(f) LayerCAM

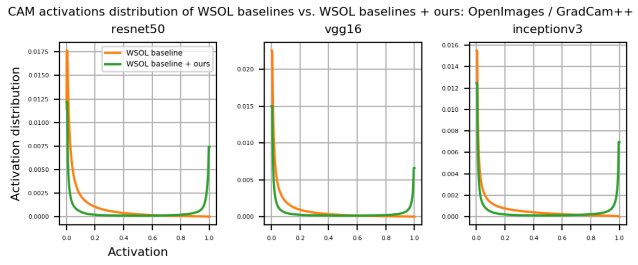
Figure 5: CAM's activation distribution over CUB test set: WSOL baselines vs. WSOL baseline + ours validated with `MaxBoxAcc`.



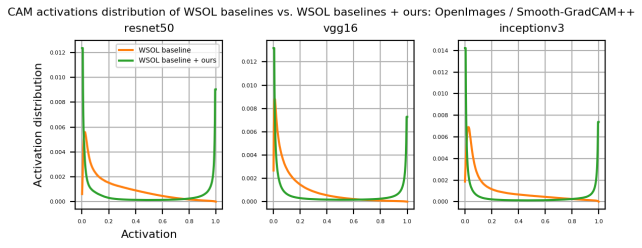
(a) CAM*



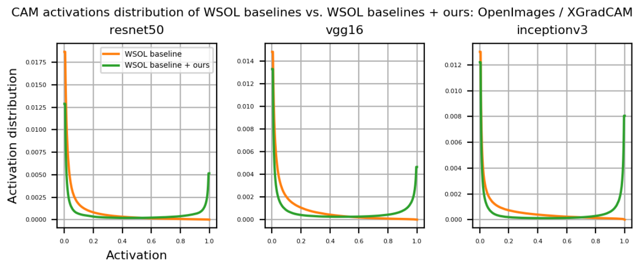
(b) GradCAM



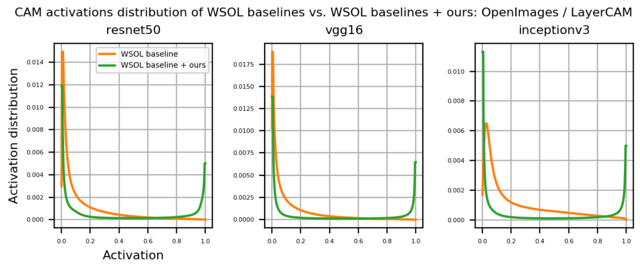
(c) GradCAM++



(d) Smooth-GradCAM++

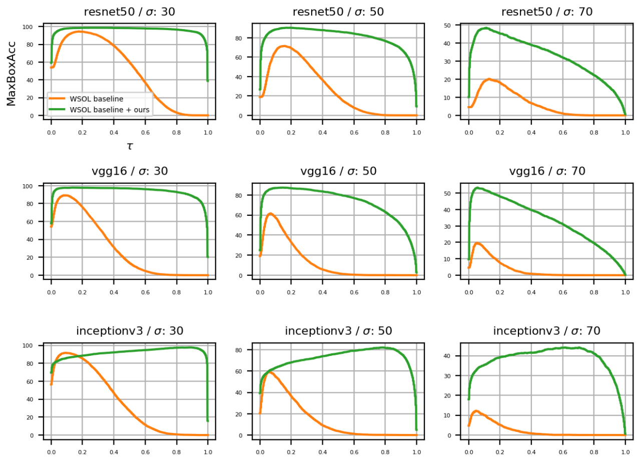


(e) XGradCAM

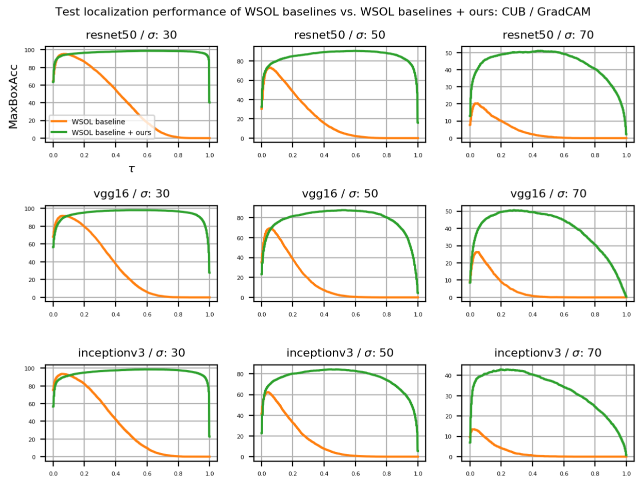


(f) LayerCAM

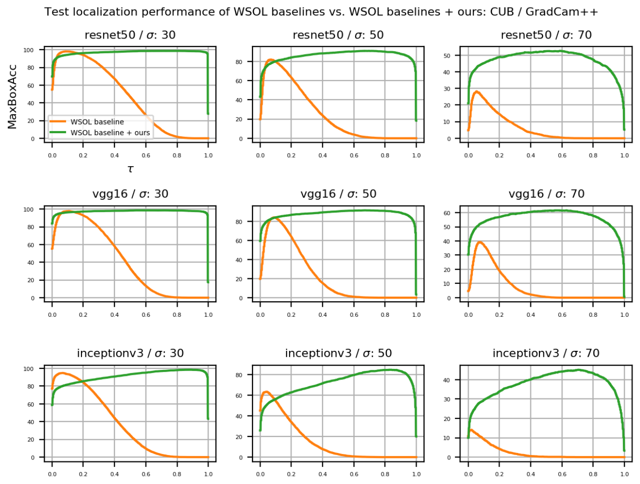
Figure 6: CAM's activation distribution over OpenImages test set: WSOL baselines vs. WSOL baseline + ours validated with `MaxBoxAcc`.



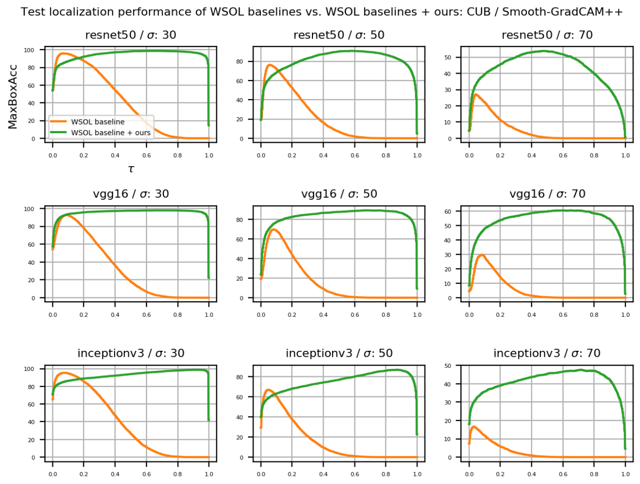
(a) CAM*



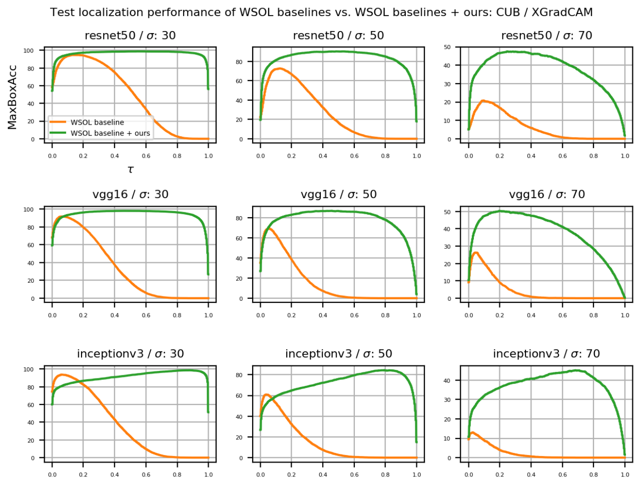
(b) GradCAM



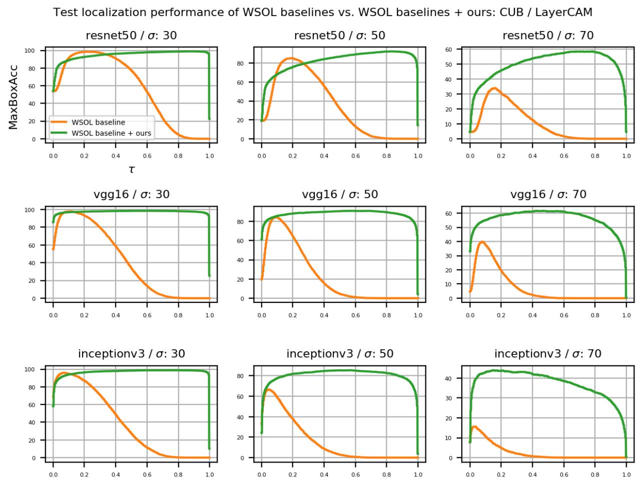
(c) GradCAM++



(d) Smooth-GradCAM++

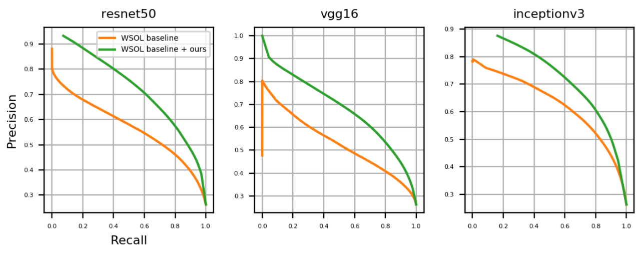


(e) XGradCAM

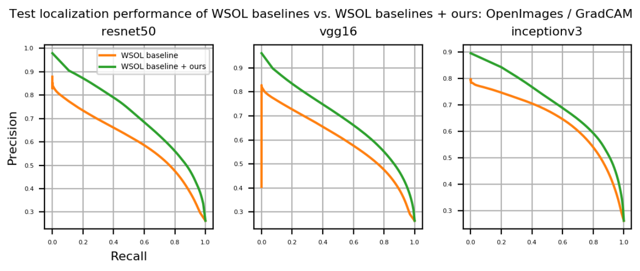


(f) LayerCAM

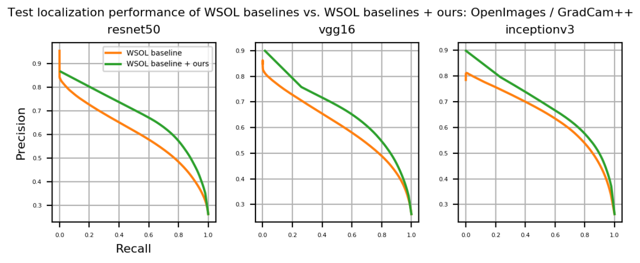
Figure 7: CUB test set: WSOL baselines vs. WSOL baselines + ours validated with `MaxBoxAcc`.



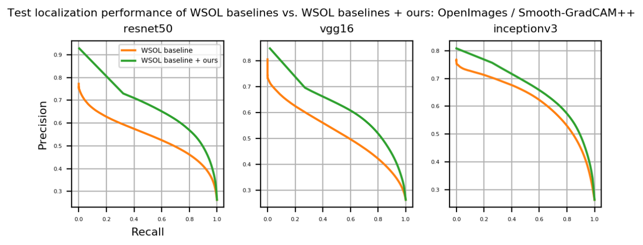
(a) CAM*



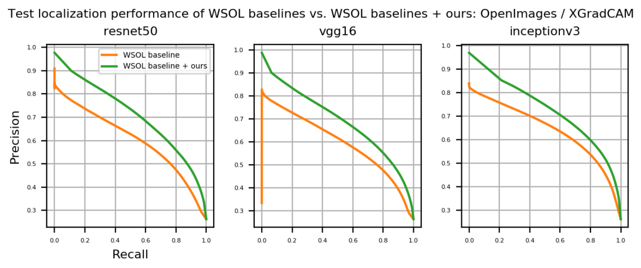
(b) GradCAM



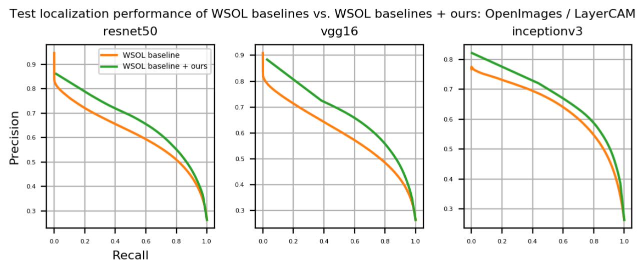
(c) GradCAM++



(d) Smooth-GradCAM++

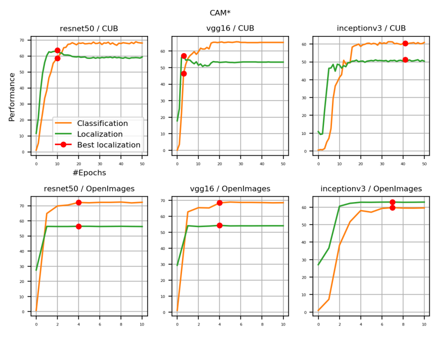


(e) XGradCAM

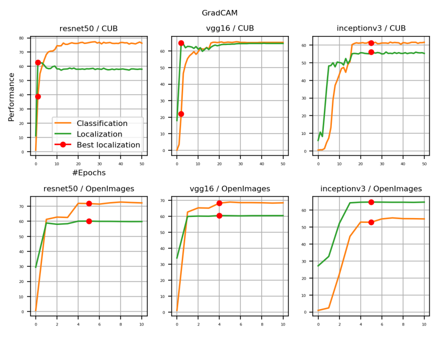


(f) LayerCAM

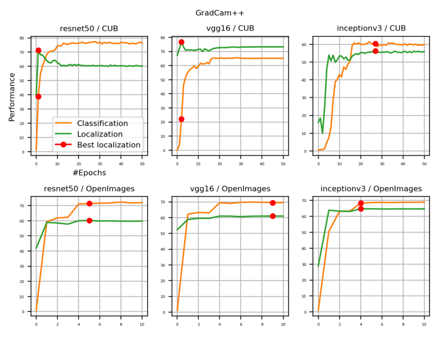
Figure 8: OpenImages test set: WSOL baselines vs. WSOL baselines + ours validated with `MaxBoxAcc`.



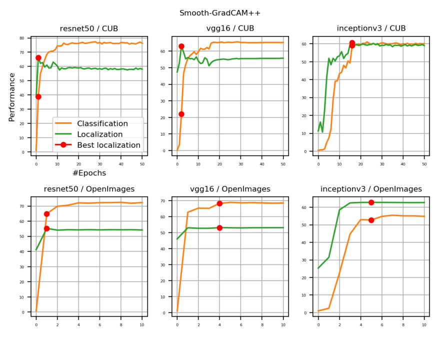
(a) CAM*



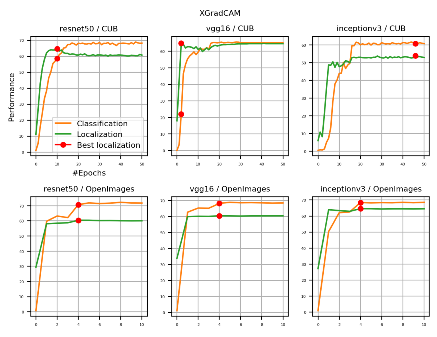
(b) GradCAM



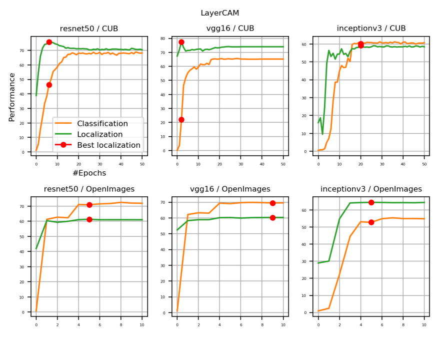
(c) GradCAM++



(d) Smooth-GradCAM++

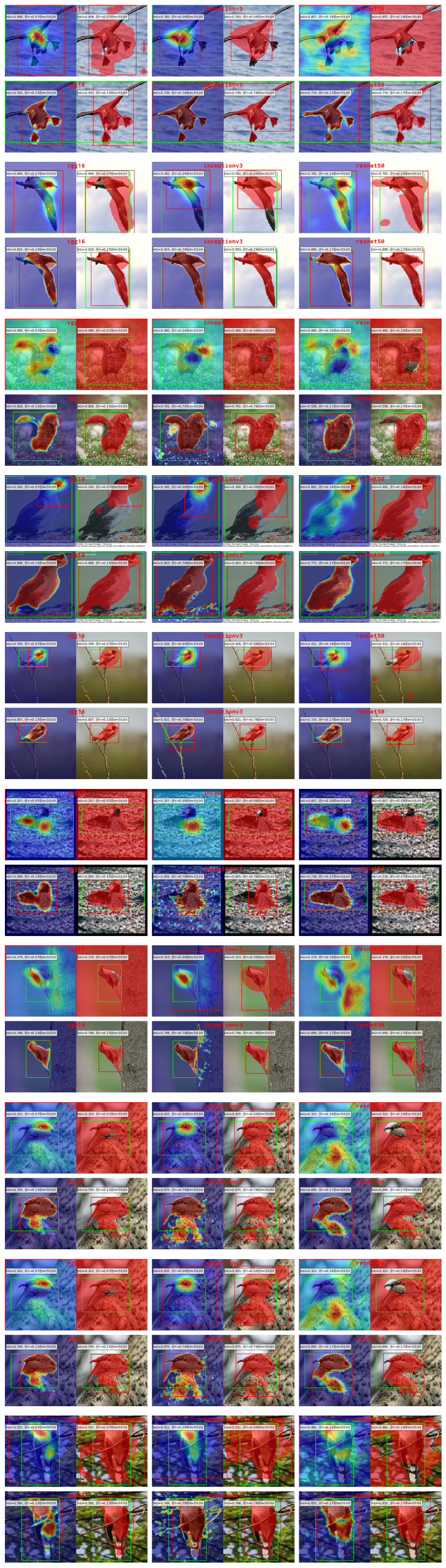


(e) XGradCAM



(f) LayerCAM

Figure 9: Convergence of classification and localization tasks over validation set over baselines WSOL methods. The red dot is the epoch for the selected model based on the localization performance using `MaxBoxAcc` metric over validation set.



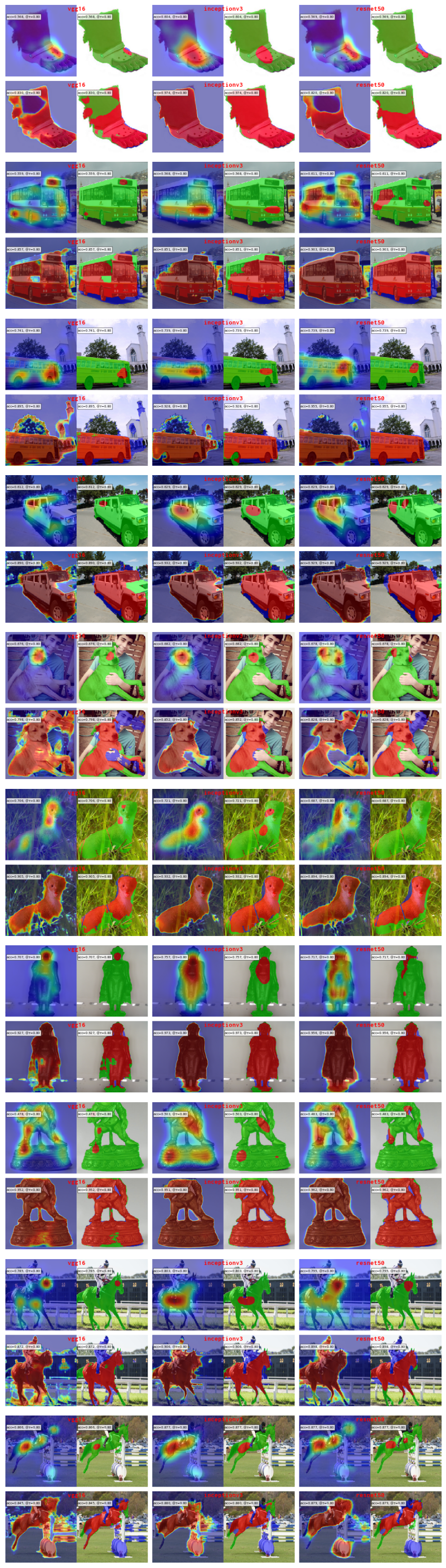
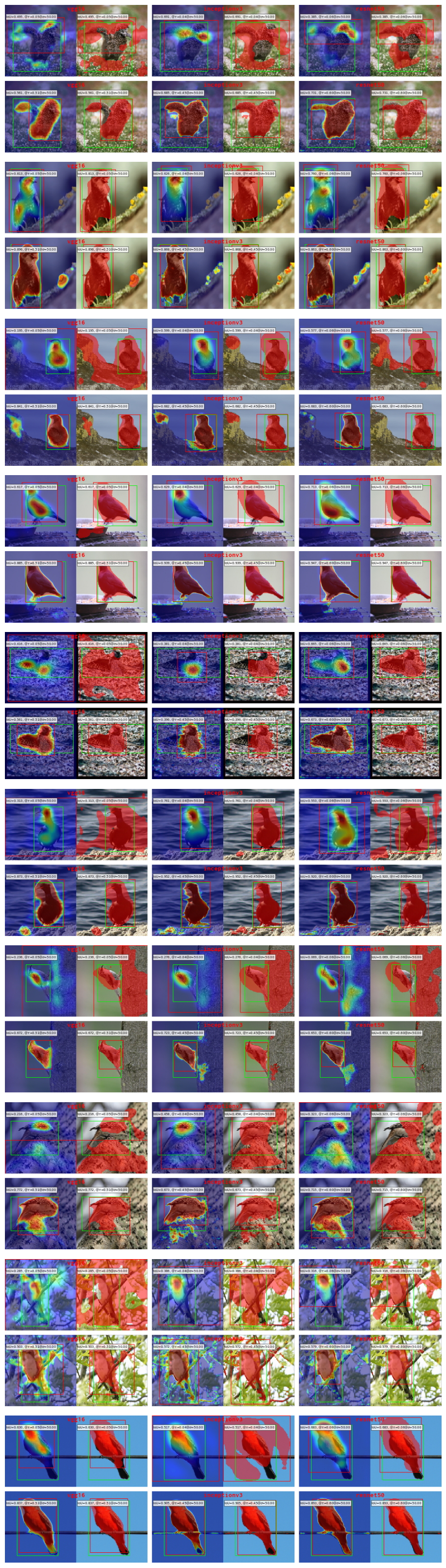


Figure 10: CAM* method examples for three backbones (left to right: VGG16, Inceptionv3, ResNet50): baselines (top) vs. baseline + ours (bottom) validated with `MaxBoxAcc`. Colors: CUB (left): green box : ground truth. red box: predicted. red mask: thresholded CAM. OpenImages (right): red mask: true positive. green mask: false negative. blue mask: false positive. $\tau = 50, \sigma = 0.8$.



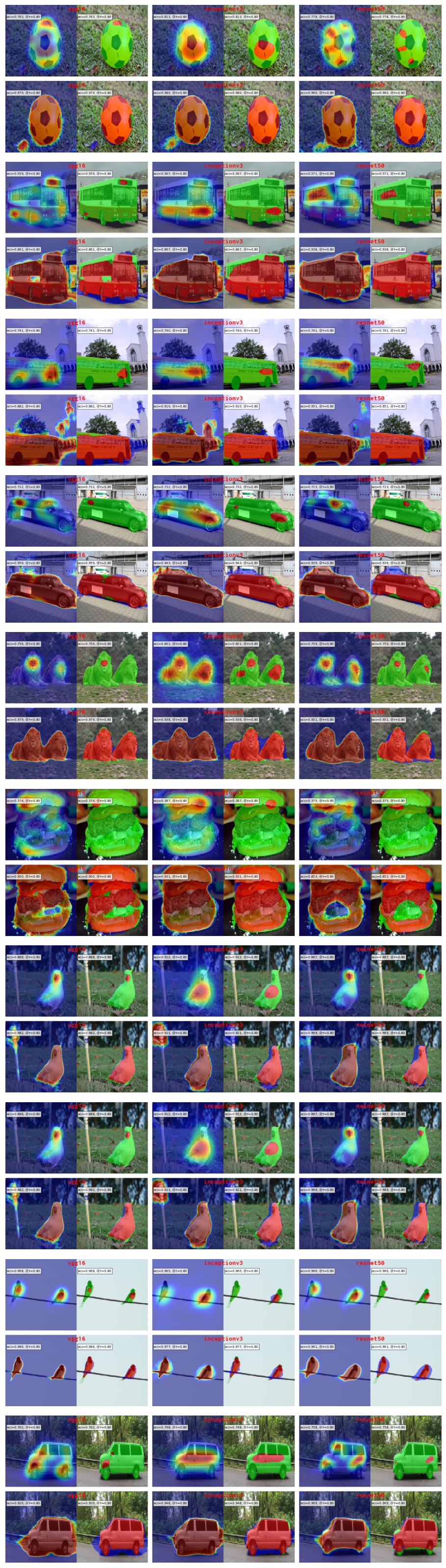


Figure 11: GradCAM method examples for three backbones (left to right: VGG16, Inceptionv3, ResNet50): baselines (top) vs. baseline + ours (bottom) validated with `MaxBoxAcc`. Colors: CUB (left): green box : ground truth. red box: predicted. red mask: thresholded CAM. OpenImages (right): red mask: true positive. green mask: false negative. blue mask: false positive. $\tau = 50, \sigma = 0.8$.

Acknowledgment: This research was supported in part by the Canadian Institutes of Health Research, Natural Sciences and Engineering Research Council of Canada, Compute Canada and MITACS.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Comput. Graph. Forum*, 29(2):753–762, 2010.

[2] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*, 2018.

[3] J. Choe and H. Shim. Attention-based dropout layer for weakly supervised object localization. In *CVPR*, 2019.

[4] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, and B. Li. Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. *BMVC*, 2020.

[5] K. He, X. Zhang, S.g Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng, and Y. Wei. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Trans. Image Process.*, 30:5875–5888, 2021.

[7] M. Ki, Y. Uh, W. Lee, and H. Byun. In-sample contrastive learning and consistent attention for weakly supervised object localization. In *ACCV*, 2020.

[8] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In J.S.-Taylor, R.S. Zemel, P.L. Bartlett, F.C. N. Pereira, and K.Q. Weinberger, editors, *NeurIPS*, pages 109–117, 2011.

[9] R. Naidu, A. Ghosh, Y. Maurya, S. R. Nayak K, and S. S. Kundu. IS-CAM: integrated score-cam for axiomatic-based explanations. *CoRR*, abs/2010.03023, 2020.

[10] R. Naidu and J. Michael. SS-CAM: smoothed score-cam for sharper visual feature localization. *CoRR*, abs/2006.14255, 2020.

[11] D. Omeiza, S. Speakman, C. Cintas, and K. Weldemariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *CoRR*, abs/1908.01224, 2019.

[12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

[13] K.K. Singh and Y.J. Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017.

[14] H. Song, M. Kim, D. Park, and J.-G. Lee. Learning from noisy labels with deep neural networks: A survey. *CoRR*, abs/2007.08199, 2020.

[15] M. Tang, F. Perazzi, A. Djelouah, I. Ben Ayed, C. Schroers, and Y. Boykov. On regularized losses for weakly-supervised cnn segmentation. In *ECCV*, 2018.

[16] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *CVPR workshop*, 2020.

[17] S. Yun, D. Han, S. Chun, S.J. Oh, Y. Yoo, and J. Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[18] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T.S. Huang. Adversarial complementary learning for weakly supervised object localization. In *CVPR*, 2018.

[19] X. Zhang, Y. Wei, G. Kang, Y. Yang, and T.S. Huang. Self-produced guidance for weakly-supervised object localization. In *ECCV*, 2018.

[20] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.