
Comprehensive Attention Self-Distillation for Weakly-Supervised Object Detection - A Reproducibility Study

Anonymous Author(s)

Affiliation

Address

email

Reproducibility Summary

1

2 **Scope of Reproducibility**

3 We perform extensive ablation studies reproducing the results of the paper “Comprehensive Attention Self-Distillation
4 for Weakly-Supervised Object Detection.” In this paper the authors propose a method of regularisation via aggregating
5 attention maps to improve the accuracy of weakly supervised object detectors. They propose to aggregate attention maps
6 from different layers of the network, and across different views of the same image, and use these maps to encourage
7 features with better object coverage. The paper claims that this allows them to train an object detector in a weakly
8 supervised manner and achieves state-of-the-art results.

9 **Methodology**

10 Using the official code released by the authors we have re-run the ablation experiments on the Pascal VOC 2007 dataset
11 using our own GPU compute resources. We made changes to the released code to further investigate the effects of other
12 model components, extending the ablation studies. We have also performed an in-depth analysis of the code itself to
13 assess code quality for ease of modification and reuse.

14 **Results**

15 We found that we were able to meet or exceed the published results. Further ablation studies showed that we were able
16 to achieve the same results without the proposed attention regularisation. Our results do not agree with the original
17 paper’s claims that the novel regularisation provides an improved object detector. We performed further ablation studies
18 to identify the source of the improvement, attributing it to the ContextLocNet-style head, Inverted Attention module,
19 regression branch, and stronger data augmentation.

20 **What was easy**

21 The fundamental concept is simple and intuitive. The figures in the paper work to make it even clearer, and provide a
22 strong motivation as to why this might be useful. Some parts of the codebase are reused from other public weakly-
23 supervised object detection codebases, anyone familiar with those works will have an easier time following this
24 one.

25 **What was difficult**

26 There was significant difficulty in working with the code due to poor coding practices, poorly documented requirements,
27 and errors in the code. Experimentation was limited due to the slow training process, which owes both to the
28 computational cost of this approach as well as some inefficient implementation choices.

29 **Communication with original authors**

30 We had no direct communication with the original authors. We browsed public communications via GitHub issues
31 posted to the public code repository.

32 **1 Introduction**

33 Weakly Supervised Object Detection attempts to train object detection models utilising only image-level class labels
34 (as opposed to the typical bounding-box level annotations). This alleviates the high annotation cost associated with
35 constructing quality object detection datasets. Since the training annotations are less granular than the desired model
36 predictions, this problem is fundamentally ill-posed and many strategies have been proposed to overcome it.

37 The authors propose that there is a degree of noise introduced into the training process due to the inconsistency between
38 model predictions on different views of the same image. In the fully supervised setting, if you train a model with
39 multi-scale inputs, flips, etc. the box annotations are transformed alongside the image. In the weakly supervised setting,
40 which typically generates box-level pseudo-annotations from its own predictions as a self-training procedure, there is
41 no guarantee that the same annotations would be selected in different scales or flips of an image. They acknowledge
42 that typically the pseudo-annotations are incomplete, and that different views often identify different instances or object
43 parts, all of which are desirable to identify. As such the authors propose to aggregate information between views
44 and across network layers into a maximal attention map, which they then use to regularise the network to encourage
45 consistent attention between views whilst also increasing object coverage.

46 **2 Methodology**

47 **2.1 Implementation(s)**

48 We utilised the published implementation by the authors for the majority of our experiments. We first performed an
49 extensive examination of the implementation to understand precise details of the method that are not discussed in the
50 paper. We found that the published code had some errors that prevent it from running as of the time of writing. We
51 fixed these issues to the best of our ability in our implementation, noting all changes in the source.

52 We also made changes for the purposes of our experiments. We instrumented the codebase with the Weights and Biases
53 platform introduced by Biewald [2020] for experiment tracking and analysis. We also added some configuration flags,
54 and utilised them to enable and disable model components by configuration for our ablation studies.

55 **2.2 Dataset**

56 We selected Pascal VOC 2007 introduced by Everingham et al. [2007] as the dataset on which to perform our ablation
57 studies. VOC07 is a popular choice for ablation studies amongst weakly supervised works as it is relatively small, and
58 so more experiments can be completed on a limited time/resource budget.

59 As is standard practice we omit annotations labelled “difficult”, and during evaluation we ignore detection of these
60 instances. We trained the model on the union of the training and validation sets (‘trainval’ containing 5011 images) and
61 evaluate on the public test set. This setting is the same as the ablation studies performed in the original paper, and is the
62 standard format used in prior weakly-supervised object detection works.

63 **2.3 Hyperparameters**

64 We use the same hyperparameters as the original paper for all experiments.

65 **2.4 Resources**

66 We utilised the two NVIDIA Titan RTX GPUs recommended by the original authors for all experiments.

67 **2.5 Model Breakdown**

68 To perform our ablation studies we need to define the individual components of the network and training procedure.

69 **2.5.1 Initial Configurations**

70 The original paper identifies eight configurations in their ablation studies. They provide a baseline, three progressively
71 more complex versions of Image-wise CASD, Layer-wise CASD, combined Image-wise and Layer-wise (i.e. ‘Full’)
72 CASD, as well as adding a regression branch and extra augmentations to the full version.

73 2.5.2 Additional Configurations

74 By analysing the code we identify two other components that are not evaluated in the ablation studies. We first note that
75 the model makes use of the pseudo-labelling algorithm introduced by Tang et al. [2018a] for all experiments. To assess
76 general applicability we also consider the simpler maximum scoring pseudo-labelling algorithm presented in Tang et al.
77 [2017].

78 We also note that rather than the multiple instance head introduced by Bilen and Vedaldi [2016] that is typical of
79 OICR-like weakly supervised object detection models, the authors utilise the ContextLocNet Contrastive-S model from
80 Kantorov et al. [2016]. This head is significantly more computationally expensive, but produces better initial seed labels
81 than WSDDN.

82 For the additional configurations we only consider their impact on the baseline, and full versions of CASD.

83 3 Code Analysis

84 An in-depth review of the code was performed to both identify any key details that are absent from the paper, as well as
85 to determine the ease of reuse and modification of the work for others that wish to reproduce the results or build on top
86 of them. It is important that research code is of high quality to ensure correctness of the results as well as reproducible
87 science.

88 3.1 Initial Setup

89 The initial setup required only some small tweaks to ensure the program launched. Some required libraries are included
90 with version information, such as Python, PyTorch, and CUDA. Others are listed as required but absent version numbers
91 mean that the most current version in the repositories may not be suitable. At time of writing this was the case with the
92 ‘scipy’ library, which is currently at version 1.7 but requires a version below 1.3. Another library is entirely absent from
93 the requirements, ‘easydict’. Finally the listed version of CUDA is in-fact incorrect, causing unexpected and somewhat
94 cryptic errors with library loading. As of time of writing the listed CUDA requirement is CUDA 10.0, however the
95 actual requirement is 9.0.

96 Although not especially difficult to debug, it would be preferable to include tested version numbers for as many libraries
97 and packages as possible. At a minimum the Python package versions should be stated for all required packages, if not
98 managed by a virtual environment package list such as pipenv or conda. It is also essential that when version numbers
99 are listed they are correct.

100 3.2 Implementation Bugs

101 There are several issues which prevent the program from running as expected or running at all. There are multiple
102 cases where arguments are passed incorrectly, either with the wrong types or in the wrong places, resulting in crashes.
103 Anyone desiring to leverage this work will have to fix these issues before being able to run the code.

104 There are also user arguments passed in when running an experiment that are overwritten by program code. The
105 provided shell script for training requires the user select a GPU by providing an index, however in the python script that
106 is subsequently launched to perform training, the selection is overwritten with the constant value “0,1” (corresponding
107 to the first two GPUs on the system). This may cause issues in multi-tenancy systems or if other experiments are
108 running, since the code forces the usage of GPUs regardless of the selection of the user.

109 Another issue with the public implementation is that Image-wise CASD is missing from the loss calculation in the latest
110 version of the public code. During the loss calculation step Image-wise CASD is calculated, however it is never added
111 to the total loss value. As such the implementation does not actually make use of it without a user modifying the code
112 to add it back into the final loss summation, along with it’s corresponding weighting factor γ .

113 3.3 Missing Components

114 The final major issue with the public code is that it does not appear to contain a functioning Layer-wise CASD
115 implementation. There is present a function `ca_lw()` however it is never called in the program and as such does not
116 contribute. Additionally the `ca_lw()` function does not appear to implement the Layer-wise CASD as described in the
117 paper, rather it performs CASD over the image and its flip more akin to an incomplete version of Image-wise CASD.

118 We made best efforts to re-implement Layer-wise CASD, however due to the complexity of the codebase and the
119 limited description in the paper we were unable to do so. As such all our experiments will lack the Layer-wise CASD
120 component.

121 3.4 General Code Quality

122 A review of the public implementation revealed several issues that make it extremely difficult to understand and modify.
123 There are several instances of no-operation or unclear lines such as `for i in range(1)` (which assigns `i` to the value
124 0 and executes the ‘loop’ once only). This adds a large cognitive overhead to working with the codebase as any potential
125 users need to interpret what these lines are actually doing and whether they have any impact on the program flow.

126 The code also has a high amount of repetitiveness. There are many cases where lines or blocks of code are repeated
127 with minor changes such as which variables are being used, index values, etc. This can make it difficult for a reader to
128 determine whether these blocks of code are doing the same thing, or how they differ, as there may only be one or two
129 characters difference between the lines, or they may even be the same with another variable having changed between
130 repeated blocks. This means that if a user wishes to modify the program (for example if they want to build on top of it)
131 they must make sure that they find all of these repeated blocks and make the same changes. Good development practice
132 would say that these repeated blocks should be either functions for those that are the same but operating on different
133 variables, or loops for those where the code is the same but an index variable has changed between repeated blocks.

134 The code also frequently switches between utilising PyTorch and NumPy for representing tensors. Both libraries
135 provide highly similar interfaces for working with tensors, however only PyTorch includes the automatic differentiation
136 components for performing backpropagation. Utilising both PyTorch and NumPy heavily can make it difficult to
137 determine which type is being used to represent a given tensor at various points in the program. As mentioned in
138 Section 3.2 the public implementation includes an error introduced by this issue, which causes the program to crash
139 when one method expecting a NumPy array is passed a PyTorch tensor instead.

140 The switching between PyTorch and NumPy also lowers the computational efficiency, as only PyTorch operates on
141 high-speed GPU devices. As such, conversions between the two involve a copy from GPU memory to system memory,
142 and then execute on the CPU, before potentially being copied again to GPU memory. These copy operations add a time
143 and space overhead that could be avoided in many cases by writing pure PyTorch code.

144 Overall the code quality issues make this work a poor choice for reuse. The code quality issues make it too easy to
145 accidentally make changes that do not do what they are intended to, or fail to make the changes uniformly across the
146 program.

147 4 Results

148 4.1 Reproducibility

Method	Original	Reproduction
Baseline	48.9	52.47
Image-wise w/o IA	52.6	52.47
Image-wise	54.1	54.12
Layer-wise	52.3	N/A
Full CASD	56.8	57.77

Table 1: Original vs Reproduced results

149 As can be seen in Table 1, the results of our reproduction mostly met or exceeded the original published results.
150 The results of the baseline and Image-wise with no Inverted Attention are identical, which was unexpected. Deeper
151 inspection of the code showed that this was because there is a flag in the code that is always set to false, disabling
152 Image-wise CASD. Regardless of this, we were able to reproduce the Image-wise results with and without Inverted
153 Attention to within ± 0.15 . Furthermore, our experiments exceeded the published result of Full CASD even though both
154 Layer-wise and Image-wise CASD are absent. These results suggest that CASD is not essential to achieving this results,
155 and may be a hindrance (although further experiments would need to be performed to establish this).

156 Due to time constraints and the slow turnaround time for experiments we were unable to re-evaluate the results after
157 the Image-wise bug was found. We were able to run some preliminary experiments which showed that the CASD
158 Image-wise loss was on the order of 10^{-5} whilst the other losses were on the order of 10^{-1} . This suggests that the

159 impact would be extremely minimal, and this is also supported by an issue on the public GitHub that was opened by
160 another user¹.

161 4.2 General Applicability

Pseudo-Labelling method	Method	mean Average Precision
OICR	Baseline	52.00
	CASD	56.73
PCL	Baseline	52.47
	CASD	57.77

Table 2: CASD with different labelling functions

162 Table 2 shows the results of using the simpler top-1 pseudo-labelling algorithm that was originally introduced with
163 Tang et al. [2017]. As can be seen there is a significant improvement regardless of the pseudo-labelling method,
164 however there is slightly more when using PCL. This suggests that the changes are generic and provide improvement
165 regardless of the pseudo-labelling method used. The use of a regression branch has been covered in prior works by
166 Ren et al. [2020], Wang et al. [2018] and is well known to improve performance even under the weakly supervised
167 setting. Stronger data augmentation is also a common practice for improving image processing models of all types,
168 and is extremely well studied. The results suggest that the Inverted Attention proposed by Huang et al. [2020a] is also
169 useful for weakly-supervised object detection.

170 4.3 Effect of ContextLocNet

Method	With	Without
Baseline	52.47	39.38
CASD	57.77	45.58

Table 3: CASD with and without ContextLocNet-style neck and head

171 As shown in Table 3 the results drop drastically with the exclusion of the ContextLocNet-style head. The difference in
172 performance between the baseline and ‘CASD’ is maintained, further strengthening the argument that the improvements
173 are agnostic to the model structure. The approximately 12 point reduction does however suggest that the overall
174 accuracy is primarily driven by the inclusion of this component instead of the simpler WSDN-style head that is used
175 in most prior works such as Ren et al. [2020], Tang et al. [2017, 2018a,b], Wang et al. [2018].

176 5 Discussion

177 By re-running the original ablation studies on our own hardware we are able to say with a high degree of confidence that
178 the overall accuracy of the work is reproducible. By utilising the author’s code it is possible to train a state-of-the-art
179 weakly supervised object detection model. However, our further studies pose questions as to whether the source of the
180 improvement is as presented.

181 The results from Section 4.1 suggest that only the Inverted Attention and flip-and-scale+colour augmentation are
182 important components in achieving strong WSOD results. This is further supported by the decision of the original
183 authors to not enable the Image-wise or Layer-wise CASD regularisation in their public implementation. Unfortunately
184 due to the state of the code discussed in 3 we missed the flag disabling Image-wise CASD, and due to time constraints
185 were unable to re-run all experiments and evaluation after fixing this bug. Anecdotal evidence from our own and others’
186 experiments as discussed supports the impact being minimal, however further experiments would need to be conducted
187 to confirm this.

188 Since the public implementation lacked CASD Layer-wise completely, and we were unable to implement it ourselves,
189 we can only speculate on its efficacy. Because we were able to exceed the published final result we can show that it
190 isn’t essential to achieving good results, however it is possible that it would provide further improvements. It is likely
191 though that similar to Image-wise CASD the loss value is extremely low. Both use the same method for computing the
192 regularisation term, just applied to different sets of attention maps.

¹<https://github.com/DeLightCMU/CASD/issues/6>

193 The results of Section 4.3 show that the methods proposed in Kantorov et al. [2016] are a major contributing factor to
194 the results. Although Kantorov et al. [2016] showed this method to be effective, only a small number of later works such
195 as Wan et al. [2019b,a] utilise it. This is most likely due to the high computational and memory cost, which includes
196 three separate passes through the fully connected layers (the most expensive parts of the network). Our results suggest
197 that it would be valuable to reconsider the inclusion of this method in modern works, as well as considering alternatives
198 that achieve similar results with a lower computational cost.

199 As discussed in Section 3 one of the largest barriers to utilising this work is the state of the public code. Improved
200 dependency management via an environment manager such as `conda` or `pipenv` would resolve most of the dependency
201 issues, being able to provide correct versions of all python libraries. Although the CUDA version issue may be
202 resolvable by `conda`, when dealing with system libraries it may be easier to manage those externally, or provide a
203 reproducible container via Docker.

204 Implementation bugs and missing components are a significant barrier to reproducible work. All authors should
205 endeavour to set up and run their public implementation in a clean environment to ensure that such bugs do not exist at
206 release. It would be ideal if these issues were caught in tests written to ensure the correctness of the released program,
207 however this is not yet standard practice and is a problem in the wider machine learning research community.

208 The general code quality issues are likely a symptom of the modern ‘publish or perish’ mindset, where researchers of all
209 levels are encouraged or required to publish as frequently as possible. This disincentives spending time on maintenance
210 tasks such as refactoring code, writing tests, etc. to ensure that it is easily utilised and reproduced. Poorly written code
211 has significant risk associated with it, as it makes mistakes more difficult to identify. We believe this is the case with
212 CASD Image-wise being disabled in the public implementation, as it was difficult to find all the required locations to
213 enable it.

214 5.1 What was easy

215 The fundamental idea of the paper is simple and intuitive. The authors explanation is clear and straightforward, with
216 diagrams to improve clarity.

217 5.2 What was difficult

218 The running of experiments was extremely slow on our Titan RTX GPU(s). Although the code allows for the usage of
219 more than one GPU, it does not do this in an efficient way and as a result receives almost no benefit from this method
220 of training. The Inverted Attention step and the ContextLocNet-style head are also both computationally expensive
221 operations to perform, this is unavoidable but contributes to this problem.

222 Experiments on the small Pascal VOC 2007 dataset took approximately 70 hours (approx. 3 days) to complete a full
223 training cycle plus a single test. To extrapolate from this, experiments on the larger VOC 2012 dataset would take
224 around 6 days and MS COCO would take over a week. In comparison, the public implementation of OICR can be
225 trained to achieve approximately 53 mAP in only 8 hours on a single GPU².

226 The code quality issues discussed in 3 also made the implementation difficult to work with. Configurations that are
227 ignored, ‘magic numbers’, unused functions and variables, and no-op lines make it a slow process to verify that any
228 changes have been made in all the appropriate locations, and are complete to take effect. This challenge presented itself
229 in this report with the absence of Image-wise CASD, since a flag is set to `false` inside the code at all times.

230 5.3 Communication with original authors

231 We did not have any direct communications with the authors. Several people (including us) have attempted to
232 communicate with the authors via the GitHub Issues on the public repository, and those issues were read and considered
233 when writing this reproducibility report.

234 6 Conclusions

235 This report sought to assess the reproducibility of the paper “Comprehensive Attention Self-Distillation for Weakly-
236 Supervised Object Detection” by Huang et al. [2020b]. We found that whilst the final results could be reproduced or
237 exceeded with the publicly released implementation, this implementation included neither a working implementation
238 of Image-wise or Layer-wise CASD as proposed in the paper. We performed extensive ablation studies to determine

²<https://github.com/ppengtang/pcl.pytorch>

239 which components contribute most to the model performance and found that the data augmentation, regression branch,
240 and utilisation of a constrastive weakly supervised head were all key factors. We also performed an analysis of the code,
241 identifying many issues that make it difficult to apply to new problems, or build on top of for future works. Overall we
242 believe that further investigation needs to be done to verify the usefulness of the Image-wise and Layer-wise CASD
243 regularisation techniques.

244 **References**

- 245 Lukas Biewald. Experiment tracking with weights and biases, 2020. URL <https://www.wandb.com/>. Software
246 available from wandb.com.
- 247 H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on*
248 *Computer Vision and Pattern Recognition*, 2016.
- 249 M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes
250 Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>,
251 2007.
- 252 Zeyi Huang, Wei Ke, and Dong Huang. Improving object detection with inverted attention. *2020 IEEE Winter*
253 *Conference on Applications of Computer Vision (WACV)*, pages 1294–1302, 2020a.
- 254 Zeyi Huang, Yang Zou, B. V. Kumar, and Dong Huang. Comprehensive attention self-distillation for weakly-supervised
255 object detection. *ArXiv*, abs/2010.12023, 2020b.
- 256 V. Kantorov, M. Oquab, Cho M., and I. Laptev. Contextlocnet: Context-aware deep network models for weakly
257 supervised localization. In *Proc. European Conference on Computer Vision (ECCV), 2016*, 2016.
- 258 Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, A. Schwing, and J. Kautz. Instance-aware,
259 context-focused, and memory-efficient weakly supervised object detection. *2020 IEEE/CVF Conference on Computer*
260 *Vision and Pattern Recognition (CVPR)*, pages 10595–10604, 2020.
- 261 Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance
262 classifier refinement. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3059–3067, 2017.
- 263 Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille. PCL: Proposal cluster
264 learning for weakly supervised object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,
265 pages 1–1, 2018a.
- 266 Peng Tang, Xinggang Wang, Angtian Wang, Yongluan Yan, Wenyu Liu, Junzhou Huang, and A. Yuille. Weakly
267 supervised region proposal network and object detection. In *ECCV*, 2018b.
- 268 Fang Wan, Chang Liu, Wei Ke, Xiangyang Ji, Jianbin Jiao, and Qixiang Ye. C-mil: Continuation multiple instance
269 learning for weakly supervised object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*,
270 pages 2194–2203, 06 2019a. doi: 10.1109/CVPR.2019.00230.
- 271 Fang Wan, Pengxu Wei, Zhenjun Han, Jianbin Jiao, and Qixiang Ye. Min-entropy latent model for weakly supervised
272 object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 02 2019b. doi:
273 10.1109/TPAMI.2019.2898858.
- 274 Jiajie Wang, Jiangchao Yao, Ya Zhang, and Rui Zhang. Collaborative learning for weakly supervised object detection.
275 In *IJCAI*, 2018.