Coreset Selection for Object Detection —Supplementary Material—

Hojun Lee[1] Suyoung Kim[1] Junhoo Lee[1] Jaeyoung Yoo[2] Nojun Kwak[1*]
[1]Seoul National University [2]NAVER WEBTOON AI
{hojun815,ksyo96,mrjunoo,nojunk}@snu.ac.kr, yoojy31@webtoonscorp.com

S1. Additional implementation details

Section 4.1 describes implementation details, and here we introduce additional implementation details. During training with the selected data, we used the SGD optimizer with a learning rate of 0.02, weight decay of 0.0001, and momentum of 0.9. The number of iterations was as follows: When the number of images was 200 or fewer, we trained the initialized network for 1000 iterations. For 500 images, we trained it for 2000 iterations. With 1000 images, we trained it for 4000 iterations. When there were 200 images or fewer, we performed training on the initialized network for 1000 iterations to ensure loss convergence. The learning rate was reduced to 0.1 times the initial value at 80% of the training iterations. The warm-up was conducted for 100 iterations, and there was no gradient clipping.

Regarding image sizes, we resized the shorter side to 800 pixels and maintained the original aspect ratio, ensuring the longer side remained below 1333 pixels. During training, resizing was done within the range of (480-800) with a step size of 32.

S2. BDD100k experiment detail

Like Pascal VOC, we experimented with Faster R-CNN-C4 with ResNet50. For the VOC experiment, we used the Faster R-CNN weight provided by detectron2. However, for the BDD experiment, there was no weight provided by detectron2. Therefore, we used ImageNet pretrained backbone to train the Faster R-CNN on the train data. We trained with a max iteration of 90k and a learning rate decay at 60k and 80k. The rest, like SGD, learning rate, and data augmentation strategy, was the same with VOC.

We initially sampled 5,000 images for each class and then ran our method. For more on this setting, please refer to Section S5. For the balance hyperparameter λ of Eq. (4), we set them to (0.075, 0.0675, 0.1, 0.1875) for the images (200, 500, 1000, 2000), respectively. After selection, we trained the selected subset during 4k iteration for 200 images, 4k iteration for 500 images, 8k iteration for 1000 images, and 8k iteration for 2000 images. And, the learning rate was decayed at 80% of the max iteration.

S3. COCO2017 experiment detail

Like Pascal VOC, we experimented with Faster R-CNN-C4 with ResNet50. Similar to VOC, we used the Faster R-CNN weight provided by detectron2 for selection.

We initially sampled 30,000 images for each class and then ran our method. For more on this setting, please refer to Section S5. For the balance hyperparameter λ of Eq. (4), we set them to (400, 800) for the images (0.05, 0.1), respectively. After selection, we trained the selected subset during 4k iteration for 400 images and 8k iteration for 800 images. And, the learning rate was decayed at 80% of the max iteration.

S4. Cross architecture detail

In the case of RetinaNet and FCOS, as mentioned in Section 4.8, following the configurations of each detector or the configurations we experimented with in Faster R-CNN resulted in significant training instability due to loss explosion. Therefore, when training with 500 images, we followed these hyperparameters: We increased the number of training iterations from 2000 to 6000, matching the learning rate decay point at 5,200 iterations accordingly. The learning rate for RetinaNet was set to 0.01, following Detectron2, while for FCOS, it was reduced from 0.01 to 0.005. We extended the warm-up iteration from 100 to 1000 iterations. Gradient clipping was introduced with a threshold of 1.0, which was previously absent. We reduced the image size in training and testing, scaling down the shorter side from 800 to 600 pixels and the longer side from 1333 to 1000 pixels. We also reduced the data resize augmentation range from 480-800 to 360-600. For other network hyperparameters, we followed Detectron2's settings for RetinaNet and used the official code's configurations for FCOS.

*Corresponding author

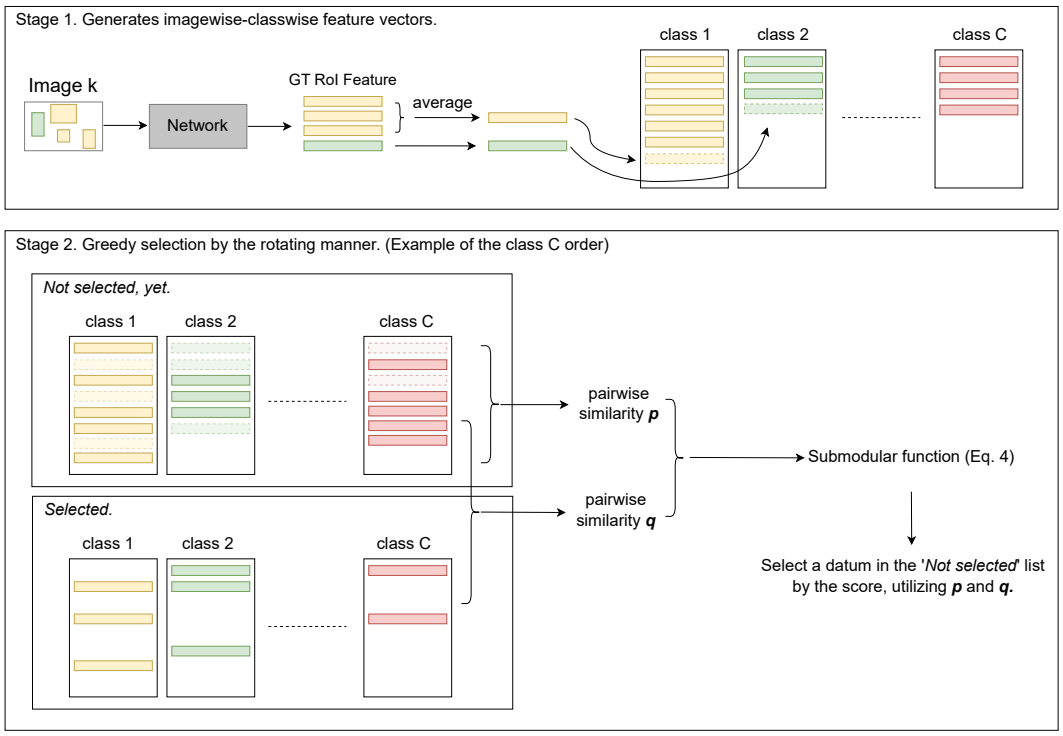

Figure S1. Stage 1 is to create imagewise-classwise feature vectors by applying RoI pooling to the ground truth boxes and subsequently averaging them by class. In Stage 2, illustrated by class C as an example, data selection for each class is performed. It calculates scores that consider the balance between previously selected and the not-selected data (Eq. 4). After selection by Eq. 4, the datum selected in the current selection step is popped from the '*Not selected, yet*' list and inserted into the '*Selected*' list. After selecting a datum for class C, the process returns to class 1 and repeats the same steps until the desired number of images is reached.

count	1e-10	0.0005	0.005	0.0125	0.025	0.0375	0.04375	0.05	0.0625	0.1	0.125	0.25	0.5	1e+10
100	22.0	21.9	30.3	34.1	34.4	34.1	34.0	33.8	33.2	33.1	33.0	33.4	32.9	33.0
200	28.8	29.2	35.0	41.5	43.8	43.8	44.3	43.5	43.3	43.3	42.0	42.7	42.1	42.1
500	44.9	44.5	46.1	48.7	52.1	53.7	53.9	53.6	54.1	53.0	52.5	52.5	52.4	52.1
1000	51.9	51.9	52.4	54.4	56.9	57.8	59.2	59.6	60.0	60.5	60.6	59.2	59.1	58.6

Table S1. AP_{50} on Pascal VOC. The optimal λ increases as the number of images grows.

S5. Sampling first, then selection for reducing cosine similarity calculation

Table S2 presents the experiment that we first reduced the number of data per class by random selection and then applied our method. Note that the numbers per class are not precisely equal, as some images might contain multiple classes, but rather approximations. We conducted this experiment for two reasons: handling classes with a large number of images, which can make cosine similarity calculations time-consuming, and addressing class imbalance.

Our results show a clear trend. Even with just 50 images per class, our algorithm outperformed random selection by a significant margin. Moreover, increasing the sample size to 1,000 images per class yielded similar performance. This implies that, even when dealing with classes with an unusually high number of images, as long as computational resources allow, sampling without replacement is expected to provide markedly better results than random selection. For reference, the minimum, maximum, and average number of images per class are 423, 6,469, and 1,281, respectively.

	Random	The number of sampled image per class							Ours
		50	100	200	300	400	500	1000	
AP_{50}	37.5	42.0	42.8	43.6	43.9	43.9	43.5	44.0	44.3

Table S2. Coreset Selection after Sampling. In all cases, we ultimately selected 200 images.

S6. Analysis of the ratio of classwise annotation counts

We explored how selection methods affect the class balance of annotations. Figure S2 shows the results. Note that in this analysis, we considered the proportions without considering the number of annotations or object sizes.

When we calculated the KL divergence with the train data, Full Random appeared to be the closest. We also evaluated how balanced the class ratios were based on the entropy. Our method showed the best balance in terms of the entropy.

Intuitively, we might expect the performance to be better when the subset's class ratios are closest to those in the train data. However, this was not necessarily the case, suggesting two possibilities. First, the number of annotations in the subsets is significantly smaller than in the train data. Therefore, having a relatively balanced dataset, even if it differs from the train data's class ratios, could be more beneficial for learning. Second, as shown in Section 4.3.1, the size of objects or the number of annotations might significantly impact performance.

S7. Coreset selection with gradients instead of RoI features

In our pursuit of selecting an informative subset for training, we conducted an experimental comparison where we explored the use of gradient vectors derived from RoI feature vectors instead of directly employing the RoI feature vectors. This approach was motivated by [1], where they demonstrated the effectiveness of gradients in image classification tasks.

The gradient vectors were obtained by backpropagating from the classification loss. Furthermore, we performed our method with the gradient of RoI feature vectors instead of RoI feature vectors. We focused solely on the gradients of classification loss because they have significantly higher dimensions than RoI feature vectors, mainly due to the multiplication of the number of classes.

In our experiments, when selecting 200 images with λ set to 0.05, we compared the AP_{50} values. Results revealed that utilizing our method with RoI feature vectors achieved an AP_{50} of 43.5, whereas when using gradients, the AP_{50} was 42.3.

S8. Class-specific hyperparameter λ set differently for each class

We experimented with setting λ of Eq. (4) differently for each class. Because the number of objects varies among classes, it leads to varying scales in the former term of the right-hand side of Eq. (4).

So, if we denote the number of objects in class c as N_c, we conducted hyperparameter tuning for λ_c, such as $\lambda_c \propto 1/N_c$, $\lambda_c \propto 1/\log_2(N_c)$, or $\lambda_c \propto 1/a^{N_c}$, where a is a hyperparameter. However, we observed no significant difference in AP_{50}. Furthermore, class-specific AP_{50} values did not reveal any consistent trends concerning the number of objects.

There are several possible explanations to consider. First, our method selects classes independently, without considering other classes. However, as mentioned in Section 1, a single image can contain multiple classes. Therefore, the assumption of complete independence among classes may not hold, and λ_c may have indirectly influenced other classes. Second, the VOC dataset exhibits a relatively less severe class imbalance compared to other datasets [3]. Based on our observations that there is not a significant performance difference within a specific range of λ values, it is possible that the imbalance is not severe.

Data imbalance is a challenging yet crucial issue addressed in many domains. Coreset selection from imbalanced data is also an area that deserves deeper exploration in the future.

S9. What would happen if we discarded all the selected images and chose again?

We observed an AP_{50} of 43.5 when we set λ to 0.05 and selected 200 images, with 10 images per class in the VOC dataset. Subsequently, we conducted an additional experiment in which we excluded the originally chosen 200 images and then selected another 200 images. In this case, the AP50 value was 42.2.

Two crucial observations emerged from these results: Firstly, the initial selection of 200 images effectively represented the entire dataset. Secondly, even when we reselected 200 images from the remaining dataset after discarding the initial selection, the performance remained significantly superior to random selection, which yielded an AP_{50} of 37.5.

S10. Measurement of CSOD's selection Time

Table S3 presents the results of measuring the time it takes for CSOD to select data. The time taken to select VOC in Section 4.2 was measured, and the time for selecting BDD in Section 4.6 was measured. As the results indicate, there is a tendency for the time to increase linearly with the number of selected images. And, comparing BDD and VOC, the more data there is, the longer it takes to make a selection. This linear increase in time is not a drawback but rather an expected and manageable aspect of the process, ensuring a thorough and proportional selection as the volume of data scales up.

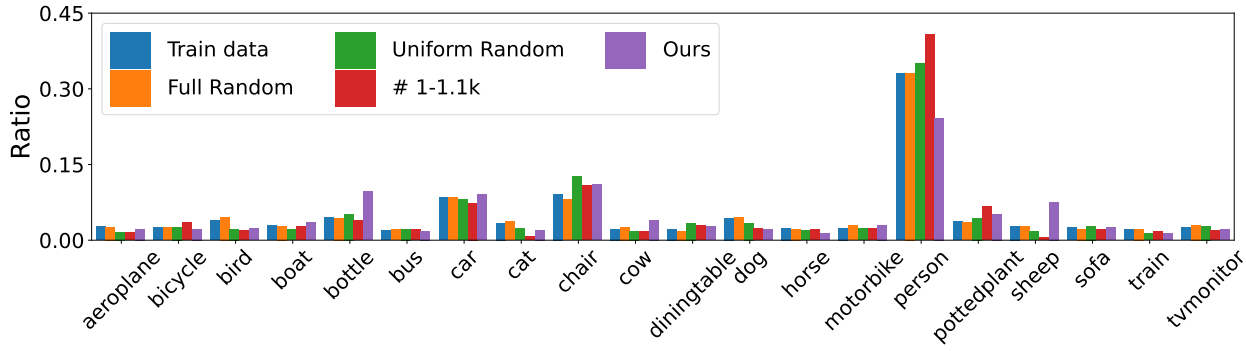

Figure S2. Annotation ratios for each class. When considering the KL divergence as the criterion with train data, Full Random was the closest. However, when calculating the balance level by computing the entropy, Ours showed the most balance.

Dataset	The number of selected images			
	200	500	1000	2000
VOC	15	42	80	158
BDD	310	712	1393	2824

Table S3. Selection time measurement result. When measuring, it was measured using CPU, not GPU. The unit is second.

S11. Comparative and adaptation study utilizing active learning for object detection

Talisman [2] applied submodular functions in the context of active learning. Talisman focuses on maximizing the performance of rare classes in active learning for object detection. Therefore, when provided with both labeled and unlabeled data, their goal is to select unlabeled data that maximizes the information about rare objects within the labeled data. While our objectives and methods are entirely different from the paper, we share a commonality in the use of submodular functions.

The main differences in the procedures of Talisman and CSOD are as follows: First, Talisman starts by randomly selecting N objects from each category, while CSOD does not follow this initial step. Second, Talisman searches for images containing at least one object that maximizes the information given the initial set of N objects. This process uses a method known as a submodular function. It's important to understand that Talisman selects these images based on the presence of a single object that greatly enriches the information content, regardless of how similar or dissimilar the other objects in the image are.

Kindly note that this succinct overview provides only a glimpse of the procedural differences, and it's crucial to recognize that the two works are guided by distinct objectives, leading to fundamentally different frameworks. Nevertheless, we adapted Talisman to our problem formulation and conducted an experiment to select 200 images from Pascal VOC.

We made several modifications to tailor the Talisman method to our needs:

- While Talisman primarily focused on rare classes, we extended its application to all classes in our problem.
- Instead of generating RoI features from unlabeled data using RPN outputs, we changed it for creating RoI features from Ground Truth (GT) boxes, given our supervised setting.
- The Talisman algorithm initially started with a few labeled rare objects. Our adaptation altered the method to start with three random images per class from the entire dataset. For a fair comparison, CSOD also started with the first three images identical to Talisman.

As a result, AP_{50} of our adaptation of Talisman was 39.3. This result was higher than the random selection of 37.5 but lower than 43.1 of ours, which started with the first three images identical to Talisman.

References

[1] Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pages 181–195. Springer, 2022.

[2] Suraj Kothawade, Saikat Ghosh, Sumit Shekhar, Yu Xiang, and Rishabh Iyer. Talisman: targeted active learning for object detection with rare classes and slices using submodular mutual information. In *European Conference on Computer Vision*, pages 1–16. Springer, 2022.

[3] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 43 (10):3388–3415, 2020.