Accurate Prediction of Bounding Boxes in Object Detection

Abstract

Object classification and localization has always been a challenging task in the field of computer vision. Accurate prediction of the bounding boxes for localization is crucial in a variety of use cases e.g. to recognize pedestrians, traffic signs, other vehicles in autonomous driving, to detect product categories in retail commerce, tumor detection in healthcare etc. These applications rely heavily on the accurate prediction of the location of the bounding boxes. Moreover, misaligned bounding box annotations throw a wrench in well-known object detection algorithms and can take significant time to diagnose and fix. Sometimes, occlusion in images due different lighting or environmental conditions pose a much greater challenge and can make it particularly hard to detect position of objects with decent accuracy. This necessitates the need for more robust models which are able to capture these common uncertainties.

In the following section, we analyse the key ideas for accurate prediction of bounding boxes in object detection as proposed by three state-of-the-art research papers given below.

1. *Bounding Box Regression with Uncertainty for Accurate Object Detection* (He et al., 2019)
2. *CornerNet: Detecting Objects as Paired Keypoints* (Law & Deng, 2018)
3. *Learning Globally Optimized Object Detector via Policy Gradient* (Rao et al., 2018)

Furthermore, we dive deeper into the implementation and results provided by the paper [1] mentioned above. The authors have provided the official implementation of the proposed algorithms in *Caffe2* deep learning framework. However, due to the recent popularity of *PyTorch* as a deep learning framework, the proposed algorithms were re-implemented in *PyTorch* which was used on top of a pre-trained backbone model *Faster-RCNN-Resnet50-FPN* (Lin et al., 2017) to get the desired results. Due to limitations in computational resources, the algorithms were evaluated on PASCAL-VOC (Everingham et al., 2010) dataset rather the large MS-COCO (Lin et al., 2014) dataset as it was much more convenient and less-time consuming. In the final section, the results from the several ablation experiments have been presented which shows that the mAP(mean average precision) values indeed match the original paper to a certain extent. We get an AP^{50} and AP^{75} value of **62.3**% and **40.2**% respectively with the KL Loss which is quite close to that reported in the original paper i.e. **57.8**% and **41.2**% respectively.

1. Literature Review

This section provides a brief summary and a critical review of three papers [1], [2] and [3]. Each of these paper present techniques for refining the position of the bounding boxes using some modification of loss function or some novel ideas in feature pooling within the convolutional layers to increase the accuracy of predictions. These ideas have been somewhat discussed in detail.

1.1. Paper I

1.1.1. Summary

This paper proposes a new regression loss function to accurately predict the bounding box locations for object detection by introducing localization variances. The method tries to address several ambiguities present in the task of object localization such as inaccurate labelling, unknown boundaries in objects due to occlusion etc. In order to capture those ambiguities, the locations of the bounding boxes are modelled as Gaussian distributions and locations of the ground truth boxes as Dirac Delta functions and the idea is to minimize the KL divergence *(Kullback-Leibler divergence)* between the two distributions (named as *KL Loss*). In addition to the locations of the candidate detections, the method also learns the variances in the localization and it uses those variances to improve on the position of the bounding boxes. Along with the *soft-NMS*(non-maximum suppression) (Bodla et al., 2017) approach used by previous state-of-the-art object detectors, the variances are used in *var-voting*(variance voting) which helps in selecting the best out of the neighboring candidate boxes. Considering the fact that the detections with high classification score does not necessarily imply better localization accuracy, *var-voting* gives higher score to detections with larger IoU and lower variance but remains invariant to their classification accuracy.

This method was experimented on two datasets MS-COCO (Lin et al., 2014) and PASCAL-VOC (Everingham et al., 2010) with several object detection architectures. The authors provided various ablation studies with VGG-16 (Liu & Deng, 2015), ResNet-50-FPN (Lin et al., 2017) as backbone and Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2017) heads on top of those backbones. By only using the KL Loss function, the method was able to improve on the average precision(AP) by 2.8%. For ResNet-50-FPN Mask R-CNN, the application of KL loss improved the AP by 1.5% and 0.9% respectively. Also combined with var-voting, it outperformed most of the state-of-the-art bounding box refinement methods with very less inference time latency addition.

1.1.2. Review

One of the useful concepts in this paper is the introduction of idea of learning the localization uncertainties which helps interpret the model better. This means that it will be more difficult to predict the bounding boxes if a model predicts larger localization variance mainly due to occlusion or some other ambiguities. Also, the variance threshold used in var-voting plays an important role as a larger threshold will take into account those candidate detections which are farther apart than a smaller threshold. Thus, it adds to the existing set of hyper-parameters and needs to be tuned appropriately for better performance based on the image context. Another important fact is that KL loss is independent of the CNN backbone so the loss can be used with any state-of-the-art object detectors without any significant computational overhead. Moreover, the idea of KL loss and var voting can be used on top of other techniques like *soft-NMS* (Bodla et al., 2017) or *IoU-guided NMS* (Jiang et al., 2018) to boost the AP performance.

1.2. Paper II

1.2.1. Summary

This paper proposes a novel approach to accurately predict bounding boxes by finding the pair of corner points, as referred to as keypoints, for a particular object in an image, especially the top-left and bottom right corner points. This approach does not use anchor boxes for location reference which have been used by several state-of-the-art methods earlier, thus reducing the complexity of the model. The location of the corner points for each object are modelled as Gaussian distributions which generate heat maps used for training. For each point, the model also learns an embedding which helps in determining the object to which these pair of corner points belong. The heat maps for the top-left and bottom-right corner points are learned separately and combined with the embeddings using the fact that if the corner points belong to the same object then the product of their embeddings is small. It incorporates two type of losses i.e. “pull” loss which is used to train the network to group the corners of the same object and the “push” loss which is used to separate the corners of different objects. The model employs an hour-glass type of convolutional network, which is originally used for human pose estimation, where the image features are down-sampled and again up-sampled along with the addition of some skip connections for capturing both localized and globalized information. It also predicts offsets to compensate for the information loss incurred while down-sampling the features. It also introduces the idea of corner pooling, a new type of pooling layer used in convolutional neural network architectures which helps in propagating localized information from the corners in a better way. The idea is to scan the feature map from right to left and perform horizontal max-pooling and from bottom to top to perform vertical max-pooling.

This method was evaluated on MS-COCO (Lin et al., 2014) dataset where it is found to outperform several state-of-the-art single stage object detectors like YOLOv2 (Redmon & Farhadi, 2017), SSD (Liu et al., 2016) etc. by an AP improvement of 42.1%. It also performs well against two-stage object detectors and outperforms methods like Mask R-CNN (He et al., 2017), Faster R-CNN (Ren et al., 2015), RetinaNet (Redmon & Farhadi, 2017) etc.

1.2.2. Review

The idea of corner pooling presented in this paper is quite helpful in the localization of bounding boxes as the corners of the bounding box generally lies somewhere outside of the body of the object. Moreover, this method does not uses anchor boxes which significantly reduces the number of hyper-parameters of the models required for training. In this paper, the size of the embedding is set as 1 during implementation which seems to work fairly descent in this case. However, it remains a question whether such an embedding size is sufficient for determining the corner points of highly occluded objects or multiple objects close together. Probably, an embedding size of 2 would better incorporate the idea of how close or far the corner points are. It is also fascinating to see that the instead of strictly modelling the corner points sharply, a Gaussian distribution was used. This would generally aid in stability of the training as the heat maps would be much more smoother.

1.3. Paper III

1.3.1. Summary

In this paper, the authors proposed a new method for globally optimizing an object detection model extending the standard cross-entropy based gradient optimization techniques involved in accurate classification and localization as used in previous object detectors. The method views the object detection task as a reinforcement learning problem where the detection network acts as an agent to find the optimal reward which in this case is the mAP(mean Average Precision) (Henderson & Ferrari, 2017) of the candidate bounding boxes with the ground truth boxes. It tries to optimize the mAP of the decision candidates and the ground truth bounding boxes given the constraint that there is a limit to the maximum number of such candidate detections. The paper uses the popular policy gradient method inspired by the *REINFORCE* algorithm (Williams, 2004) where the detection candidates are jointly processed taking into account the overall global information. Compared to this method, the previous state-of-the-art object detectors used only localized information by optimizing each candidate independently in addition to using various other post processing tricks to find the best detection candidates. To reduce the variance in the policy gradient estimation, the method computes the loss against some standard baseline which in this case is considered to be the mAP obtained by only utilising the standard NMS approach (Luo et al., 2021). Furthermore, a component of cross entropy loss is added to stabilize the training along with some regularization terms.

1.3.2. Review

The method used in this paper considers the object detection model as a reinforcement learning agent which has a finite set of actions to select candidate detections in order to optimize the mAP. Ideally, there is a large number of such actions that the model can choose while finding the optimum. However, the authors propose that sampling a small number of such actions in a single gradient step is enough to provide significant stability and efficiency. For the experiments, the authors have sampled $k = 10$ actions. Even though such action spaces are large and computationally expensive for exploration, no study or comparison was provided on the stability of the algorithm if the number of such actions sampled in each step were varied. Also, since this method does not depend on the architecture of the CNN backbone, it is evident to work with any object detection models.

2. Implementation

This section presents an overview of the motivation and steps to implement the methods presented in the paper

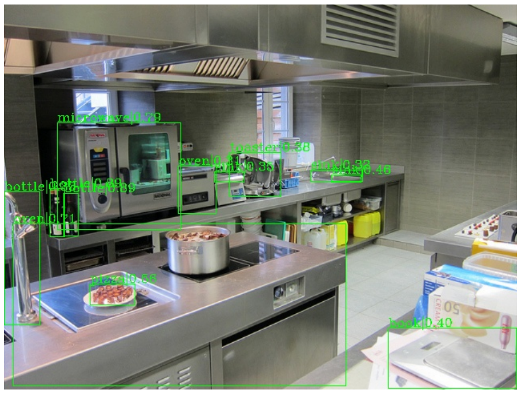


Figure 1. Predicted bounding boxes on a sample MS-COCO image using *KL-Loss-pytorch MMDetection* pretrained model with KL Loss

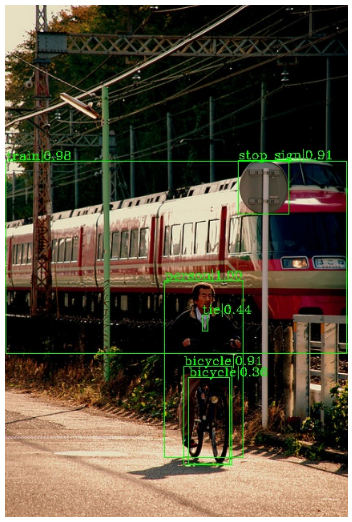


Figure 2. Predicted bounding boxes on a sample MS-COCO image using *KL-Loss-pytorch MMDetection* pretrained model with KL Loss

"Bounding Box Regression with Uncertainty for Accurate Object Detection" (He et al., 2019) and share some insight into some of the experimental results.

2.1. Motivation

In the paper, the authors have provided the official implementation of the proposed loss function called KL Loss and the algorithm for *var-voting* as a a part Detectron, which is Facebook's AI research software system for object detection and semantic segmentation. This system is built in Python with the support of Caffe2 deep learning framework. However, more recently Facebook has released Detectron2, which is an upgrade to its previous system which is supported by the popular PyTorch deep learning framework. Through considerable research efforts, it was found that

model	AP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR^{S}	AR^{M}	AR^{L}
baseline(Without KL Loss)	0.317	0.587	0.305	0.263	0.373	0.36	0.345	0.416	0.413
KL-Loss	0.471	0.697	0.503	0.419	0.617	0.541	0.469	0.65	0.637
KL-Loss + var-voting	0.477	0.7	0.525	0.427	0.637	0.527	0.476	0.669	0.615
KL-Loss + var-voting + soft-NMS	0.49	0.702	0.558	0.41	0.642	0.557	0.464	0.69	0.7

Table 1. Comparison of different methods using *KL-Loss-pytorch* model pretrained on MS-COCO. The baseline model is ResNet-50-FPN Faster R-CNN with 91 categories. Inference was done on a small subset of MS-COCO images to get the desired mAP values.

there are no official implementations present for KL Loss in Detectron2 or other popular object detection AI frameworks (e.g. OpenMMlab's MMDetection) with the exception of a couple of scholarly repositories given below.

- KL-Loss (Official, Caffe2, Detectron)
- Stronger-Yolo-pytorch (PyTorch)
- KL-Loss-pytorch (PyTorch, MMDetection)

The only trusted implementation based on PyTorch which has been provided by the author himself is the inclusion of KL Loss in the implementation of Yolo-V3 (Stronger-YOLO) (Redmon & Farhadi, 2018) model. Since the method used by the author in the paper is model agnostic, hence the implementation of the loss function and the *var-voting* can be done in a more generalized fashion to be used with any state-of-the-art object detector using PyTorch. Thus, in this project, the two approaches were implemented in PyTorch and tested using a pretrained object detector backbone model so that those methods can be later incorporated in the Detectron2 system and be widely available for use to the entire AI community. Since, Detectron2 is a configuration based framework for training and inference, creating a separate module for KL-Loss function will enable switching between the loss functions much easily. Moreover, a simple modification of object detector's loss to use the KL loss, REINFORCE type loss((Rao et al., 2018)) or any other type of loss function during training of any other similar models like *Faster-RCNN*, *CornerNet* (Law & Deng, 2018) etc. can aid in capturing the variability present in the detected key-point locations and can draw parallels between the original and modified approaches.

In this process, it is evident that the implementation of KL Loss function will generate the idea of mathematically modelling losses using probabilistic functions other than Gaussian which can better capturing the variability in the predictions of the bounding box locations. Also, it will enable the use of such loss functions in any object detection model in a generalized way with minimal changes in the model architecture. Even if an object detector is trained on an object detection dataset devoid of severe occlusion or mismatched labels, it can be made suitable to perform better on datasets with increased unpredictability and occlusion by fine tuning the model and incorporation of the KL loss. Comparing the models based on different loss functions will provide more visibility and will have much more interpretability.

2.2. Implementation Steps

Firstly, the idea was to find and run any existing PyTorch implementation of popular state-of-the-art object detection models to get a feel of the structure on how the object detection is done on different types of datasets like MS-COCO(Lin et al., 2014), Pascal-VOC(Everingham et al., 2010) etc. Carrying out inference on some sample images using the readily available pretrained models was a good starting point as it gave an some insight into the annotation formats prevalent in object detection literature. Some sample inferences were performed on different types of datasets to compare the classification scores at high level.

Below are the steps performed to implement the proposed approaches and reproduce some of the experiments presented in the paper[1].

2.2.1. Data Setup

The paper uses two object detection datasets MS-COCO(Lin et al., 2014) and PASCAL-VOC(Everingham et al., 2010) for the training and bench-marking the methods. MS-COCO is a large dataset which has several versions. The 2017 version of the dataset has 118K training images ($\sim 18GB$) with 80 classes, 5K validation images ($\sim 1GB$) and 41K test images($\sim 6GB$). PASCAL-VOC(2007) is smaller compared to MS-COCO with 20 object classes having 2.5K training images($\sim 439MB$) and 2.5K validation and testing images($\sim 430MB$). Since the format of these datasets generally vary, it is more appropriate and easier to stick to one format like COCO format having json-based annotations. The pre-trained backbone models are generally trained on MS-COCO and hence to perform experiments with datasets like PASCAL-VOC, KITTI(Geiger et al., 2013) etc. it is better to convert them to COCO format and train the model using that format. For the current implementation, PASCAL-VOC dataset, which is small and relevant for functional testing of the algorithms, was used with the goal to eventually move towards the larger datasets like MS-COCO. A custom pytorch dataset implementation was added to handle the necessary format conversion and pre-

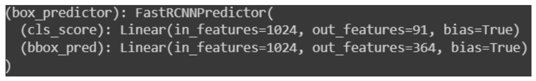
model	AP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR^{1}	AR^{10}	AR^{100}
baseline (Without KL Loss)	0.391	0.654	0.420	0.167	0.401	0.501	0.360	0.545	0.566
KL-Loss	0.372	0.623	0.402	0.159	0.385	0.477	0.352	0.539	0.558
KL-Loss + var-voting	0.394	0.707	0.347	0.331	0.403	0.421	0.308	0.451	0.458
KL-Loss + var-voting + soft-NMS	0.413	0.715	0.363	0.345	0.424	0.426	0.308	0.537	0.549

Table 2. Comparison of different methods using the implemented model trained on PASCAL-VOC. The baseline model is ResNet-50-FPN Faster R-CNN with 21 categories. Inference was done on a subset of PASCAL-VOC validation images to get the desired mAP values.

processing required before feeding the data into the model.

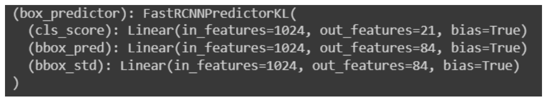
2.2.2. Initial Experiments

In the initial phase, some preliminary experiments were performed with *"KL-Loss-pytorch"* repository [2.1] based on the OpenMMLab's MMDetection framework which has reproduced the results. For testing purposes, inference was run on a small subset of COCO dataset containing 16 randomly sampled images. The detection results are show in Fig.1 and Fig.2. For inference, the model was loaded with pre-trained weights already made available in the repository. The model uses a ResNet-50-FPN backbone with Faster-RCNN trained on MS-COCO dataset with KL-Loss.



```
(box_predictor): FastRCNNPredictor(
  (cls_score): Linear(in_features=1024, out_features=91, bias=True)
  (bbox_pred): Linear(in_features=1024, out_features=364, bias=True)
)
```

Figure 3. Original *box_predictor* module of *Faster-RCNN-ResNet50-FPN* with 91 categories



```
(box_predictor): FastRCNNPredictorKL(
  (cls_score): Linear(in_features=1024, out_features=21, bias=True)
  (bbox_pred): Linear(in_features=1024, out_features=84, bias=True)
  (bbox_std): Linear(in_features=1024, out_features=84, bias=True)
)
```

Figure 4. Modified *box_predictor* module of *Faster-RCNN-ResNet50-FPN* where *bbox_std* was added as an output of a fully connected layer for localization variance prediction with 21 categories

2.2.3. Model Selection

Since the paper uses two types of backbone architectures i.e. VGG-16 and ResNet-50-FPN, the idea was to try with one architecture. The ResNet-50-FPN architecture is readily available in PyTorch and other popular object detection repositories have generally used the same. So, the model ResNet-50-FPN pretrained on MS-COCO dataset was chosen as the backbone as it provides a good baseline for subsequent experiments. However, the ResNet-50-FPN is quite a deep architecture having significantly large number of parameters than VGG-16. Given the limitation on the access of good GPU resources, an option was always kept at hand to resort to VGG-16 as backbone if required.

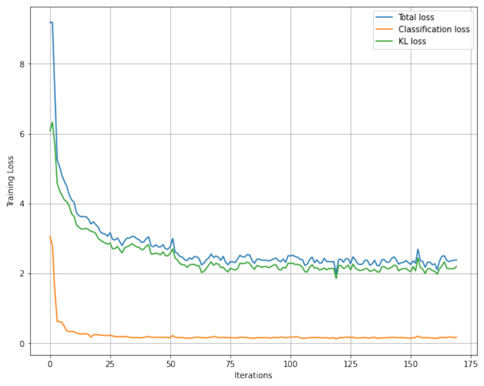


Figure 5. Convergence of KL Loss

2.2.4. Algorithm Implementation

As part of the experimentation, the algorithm for computing KL Loss and the algorithms for *var-voting* and *soft-NMS* were implemented in PyTorch keeping the original Caffe2 implementation as reference. Fortunately, the backbone architecture Faster-RCNN-ResNet-50-FPN was readily available in PyTorch model repository(Model Zoo), so no significant implementation activities were required in that front. However, the **RoI Box Head** module of the model was modified to get the predicted localization variances of the candidate detections which were used in computation of KL-Loss and var-voting. If x_g and x_e represent the ground truth and expected coordinates, then the KL loss is given as follows:

$$L_{reg} = \begin{cases} e^{-\alpha}(|x_g - x_e| - \frac{1}{2}) + \frac{1}{2}\alpha & \text{if } |x_g - x_e| > 1 \\ \frac{e^{-\alpha}}{2}(x_g - x_e)^2 + \frac{1}{2}\alpha & \text{otherwise} \end{cases} \tag{1}$$

Here, the network outputs the variances as $\alpha = log(\sigma^2)$ to avoid the problem of exploding gradients. The algorithms for *var-voting* and *soft-NMS* were added in the testing mode of the model as a part of post-processing the candidate bounding boxes. Also, to accommodate for the lesser number of classes in PASCAL-VOC dataset i.e. 21 object categories(including background) in comparison to 91 categories in MS-COCO, the output dimension in final inference layer was set as 21 as shown in Fig. 3 and Fig. 4.

2.2.5. Training & Evaluation

For training the modified model, all the hyper-parameters were kept same as that in the original paper and trained for a small number of epochs (e.g. 10 or 25 epochs) on the PASCAL-VOC training dataset. The backbone model weights were unchanged and were set as non-trainable during training. The evaluation was be performed using the metric *"mean Average Precision"(mAP)* (Henderson & Ferrari, 2017). An already available mAP(Cartucho et al., 2018) implementation from Python's *torchmetrics* library was used to measure the performance on the PASCAL-VOC validation dataset. The mAP values were observed for different model configurations.

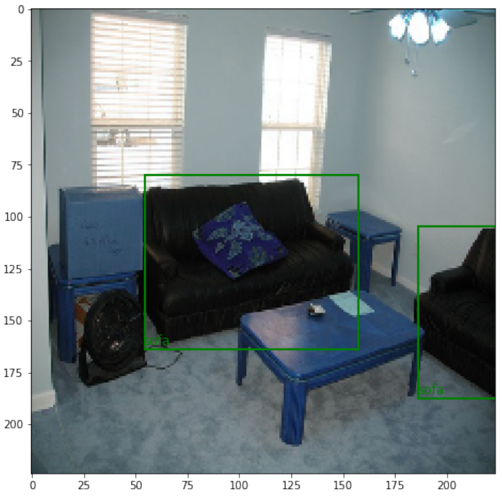


Figure 6. Ground truth bounding boxes in a sample PASCAL-VOC image

3. Results

Using the *"KL-Loss-pytorch"* repository [2.1] implementation, the mAP results were generated on a small subset of MS-COCO dataset containing 16 random images. The results are shown in Table 1. These results provide a baseline for comparing the ones obtained using the custom implementation of the model. Furthermore, the custom implementation was trained using the KL loss function on the PASCAL-VOC training dataset. Some of the sample detections on the validation data are shown in Fig. 7 and Fig. 8, given the ground truth boxes as shown in Fig. 6. Ablation experiments were performed with several different configuration - (1) Without KL Loss, (2) With only KL Loss, (3) With KL Loss + Variance Voting and (4) With KL Loss + Variance Voting + Soft-NMS. The results are shown in Table 2. As can be seen, with KL Loss + *var-voting*+*soft-NMS*, there is a significant rise in the mAP value in comparison to the model trained with only KL-Loss which gives an indication that the proposed refinement approaches based on

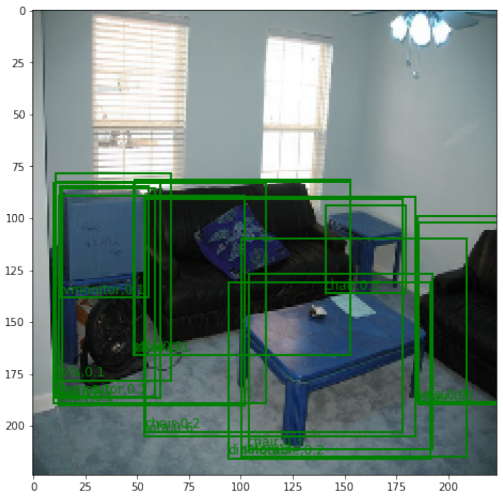


Figure 7. Predicted bounding boxes on a sample PASCAL-VOC image using the proposed implementation without KL Loss. Number of boxes predicted is 16. Top two predictions are 'sofa' having confidence score 0.94 and 0.92 respectively.

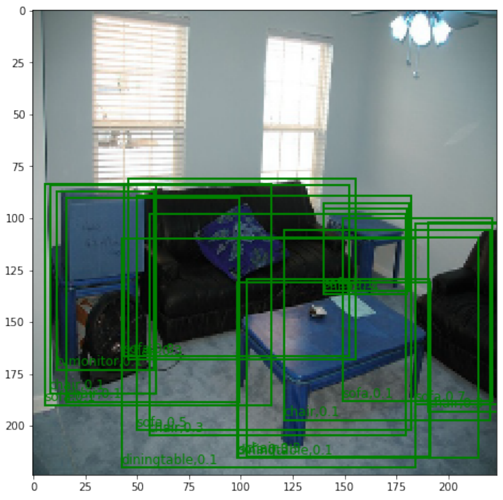


Figure 8. Predicted bounding boxes on a sample PASCAL-VOC image using the proposed implementation with KL Loss. Number of boxes predicted is 18. Top two predictions are 'sofa' having confidence score 0.76 and 0.73 respectively.

the localization variances indeed perform better. However, since the custom model was trained only on a small dataset the mAP values are less than that from the pretrained model from the repository [2.1]. But usage of *var-voting* and *soft-NMS* on top has some positive impact on the mAP values. Also the custom model did not perform well in detecting smaller objects as evident from the AP^S value of **15.9**% in KL-Loss implementation as compared to **41.9**%. This can be attributed generally due to the absence of ground truth boxes of smaller objects in the dataset. Also, the average number of detections after refinement was found to be similar to the baseline model which indicates that KL Loss is not intervening with the detection process rather it is producing a better approximation for the box coordinates.

4. Conclusion

In conclusion, the mAP results obtained using the custom implementation of KL-Loss and *var-voting* seems to ascertain the fact that better refinement in bounding box locations can be performed using these approaches. The mAP values can be increased by significantly using the proposed methods. Overall, this implementation was able to achieve decent performance based on the limited training and data used. During the experiments, there were several challenges that were overcome by carefully handling the datasets and compute resources. Some of the issues which were faced during the process are as follows:

- In some cases, the datasets were not in the correct format and customization was required to get the images and annotations loaded into the dataloader. In other cases, the training data had some invalid annotations for some of the images. It is thus essential to check for any source of errors present due to invalid annotations before proceeding towards the training phase.

- As in case of MS-COCO, the dataset is too large and it consumed considerable GPU resources while training. Frequent timeouts of Google Colaboratory notebooks in cloud as well as local machines occurred in several scenarios and GPU memory overflow errors were even encountered. To tackle this problem, PASCAL-VOC dataset was used and the intention was to use a smaller subset of the COCO dataset ($\sim 20\%$ of the entire set).

- During some preliminary training, the training stopped several times at $loss.backward()$ step. This is due to too much GPU memory usage which caused the program to crash. However, the issue was resolved by setting a smaller batch size for training. Also making a larger chunk of the parameters as non-trainable, gradient calculations were minimized for quick training and evaluation.

- Another issue encountered was the mismatch in the dimensions and channel ordering of the images while feeding into the network. By careful debugging, this issue can be rooted out as well.

A significant effort has been made towards running the experiments in as simple way as possible. The goal was to include this loss and the algorithm for *var-voting* as a configuration in the Detectron2 framework making it easily controllable so that those methods can be used based on the context. Since the repositories of Detectron and MMDetection are quite complicated, it took considerable amount of hours in understanding the structure of the repositories and the associated configuration which can be controlled to get the corresponding model architecture and hyper-parameters used for training and testing. This urged me to simplify the method and use smaller models. The code repository based on OpenMMLab's MMDetection framework also contains some relevant modules for bridging the gap between a PyTorch based implementation to configurations for easier accessibility. Hence, some ideas from that repository were beneficial in the process. Though the modules for KL-loss and *var-voting* could not be directly integrated to the Detectron2 framework but the methods have been made modular enough to be directly used without any additional overhead.

References

Bodla, N., Singh, B., Chellappa, R., and Davis, L. S. Soft-nms - improving object detection with one line of code. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5562–5570, 2017. URL http://arxiv.org/abs/1704.04503.

Cartucho, J., Ventura, R., and Veloso, M. Robust object recognition through symbiotic deep learning in mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2336–2341, 2018. URL https://doi.org/10.1109/IROS.2018.8594067.

Everingham, M., Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2): 303–338, jun 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4. URL https://doi.org/10.1007/s11263-009-0275-4.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. URL https://doi.org/10.1177/0278364913491297.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017. doi: 10.1109/ICCV.2017.322. URL https://doi.org/10.1109/ICCV.2017.322.

He, Y., Zhu, C., Wang, J., Savvides, M., and Zhang, X. Bounding box regression with uncertainty for accurate object detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2883–2892, 2019. doi: 10.1109/CVPR.2019.00300. URL https://doi.org/10.1109/CVPR.2019.00300.

Henderson, P. and Ferrari, V. End-to-end training of object class detectors for mean average precision. In *Computer*

Vision – ACCV 2016, Lecture Notes in Computer Science, pp. 198–213. Springer, Cham, March 2017. ISBN 978-3-319-54192-1. doi: 10.1007/978-3-319-54193-8_13. URL http://www.accv2016.org/. 13th Asian Conference on Computer Vision, ACCV'16 ; Conference date: 20-11-2016 Through 24-11-2016.

Jiang, B., Luo, R., Mao, J., Xiao, T., and Jiang, Y. Acquisition of localization confidence for accurate object detection. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision – ECCV 2018*, pp. 816–832, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01264-9. URL http://arxiv.org/abs/1807.11590.

Law, H. and Deng, J. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. URL http://arxiv.org/abs/1808.01244.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. URL http://arxiv.org/abs/1405.0312.

Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017. URL http://arxiv.org/abs/1612.03144.

Liu, S. and Deng, W. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734, 2015. doi: 10.1109/ACPR.2015.7486599. URL https://doi.org/10.1109/ACPR.2015.7486599.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision – ECCV 2016*, pp. 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0. URL http://arxiv.org/abs/1512.02325.

Luo, Z., Fang, Z., Zheng, S., Wang, Y., and Fu, Y. Nms-loss: Learning with non-maximum suppression for crowded pedestrian detection. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, ICMR '21, pp. 481–485, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384636. doi: 10.1145/3460426.3463588. URL https://doi.org/10.1145/3460426.3463588.

Rao, Y., Lin, D., Lu, J., and Zhou, J. Learning globally optimized object detector via policy gradient. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6190–6198, 2018. doi: 10.1109/CVPR.2018.00648. URL https://doi.org/10.1109/CVPR.2018.00648.

Redmon, J. and Farhadi, A. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017. doi: 10.1109/CVPR.2017.690. URL https://doi.org/10.1109/CVPR.2017.690.

Redmon, J. and Farhadi, A. Yolov3: An incremental improvement, 2018. URL http://arxiv.org/abs/1804.02767. cite arxiv:1804.02767Comment: Tech Report.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004. URL https://doi.org/10.1007/BF00992696.