Neighborhood Sampling Confidence Metric for Object Detection

Christophe Gouguenheim, Ahmad Berjaoui

Neighborhood Sampling Confidence Metric for Object Detection

Christophe Gouguenheim[1,2] **Ahmad Berjaoui,**[3,2]

[1] Thales Alenia Space
[2] IRT SystemX
[3] IRT Saint-Exupery
christophe.gouguenheim@thalesaleniaspace.com, ahmad.berjaoui@irt-saintexupery.com

Abstract

Object detection using deep learning has recently gained significant attention due to its impressive results in a variety of applications, such as autonomous vehicles, surveillance, and image and video analysis. State-of-the-art models, such as YOLO, Faster-RCNN, and SSD, have achieved impressive performance on various benchmarks. However, it is crucial to ensure that the results produced by deep learning models are trustworthy, as they can have serious consequences, especially in an industrial context. In this paper, we introduce a novel confidence metric for object detection using neighborhood sampling. We evaluate our approach on MS-COCO and demonstrate that it significantly improves the trustworthiness of deep learning models for object detection. We also compare our approach against attribution-guided neighborhood sampling and show that such a heuristic does not yield better results.

Introduction

Recent advances in object detection have made it attractive to use in industrial use-cases, for example for finding flaws in manufactured motherboards, or for finding targets of interest in satellite images (Lam et al. 2018). In most cases, a major roadblock for the adoption of deep learning based image detection in an industrial context is the inability to assess the trustworthiness of network predictions.

A naive approach relies on the probability associated with the highest ranking class in the output along with the bounding box around the detected object. However, it is well-known that this value is not representative of the quality of the model prediction (Szegedy et al. 2013).

In this paper, we propose a novel confidence metric based on neighborhood sampling (NS), whereby we test the conformance of the predictions for samples picked randomly in the neighborhood of the input.

There are several advantages to this method:

- It is black-box, i.e. it does not depend on a particular model architecture, and by extension to its particular weights. The only assumption we make is about the shape of the model's output. In particular, this makes our method compatible with most architectures for object detection neural networks FasterR-CNN (Ren et al. 2015), Yolo (Redmon et al. 2016) and SSD (Liu et al. 2016).
- It is easy to implement. This is mainly because our experiments showed that, on selected use-cases, the use of attributions to pre-select samples had no real impact on overall performance of the confidence metric. This allowed us to focus on a naive, therefore simple implementation of neighborhood sampling, and test its performance.

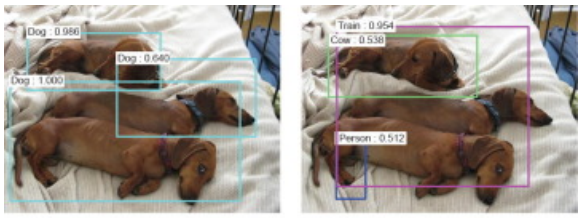

Figure 1: A picture generated using an adversarial attack produce wrong predictions with a high level of confidence (Ren et al. 2020)

Given the fact that we use vanilla neighborhood sampling, without *intelligently* selecting image pixels to turn-off (as done in (Jha et al. 2019)), we put special focus on experimentation to validate this NS confidence metric.

Our experiments have several goals:

1. Assess the validity of the method to be used as a confidence metric.
2. Tune the hyper-parameters of the method, and especially the sample size, which have a direct impact on metric performance.
3. Validate the stability of hyper-parameters across classes in a single use-case, and across use-cases.

In summary, we make the following contributions in this paper:

- Provide a new and simple confidence score for object detection
- Prove that using attributions to accelerate neighborhood sampling is not necessary in practice.

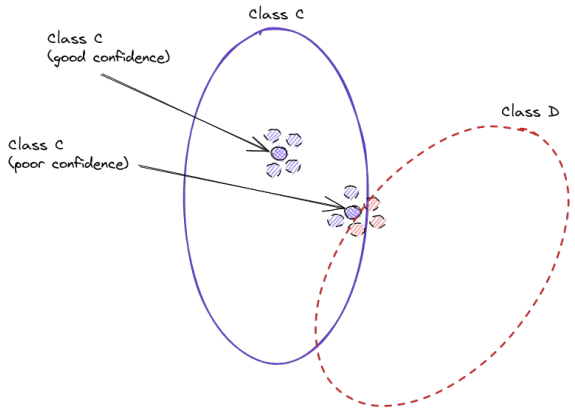

Figure 2: Neighborhood sampling (NS) allows us to test the distance of our input to the border with another class. The closer to the border, the less confidence we have in the prediction.

Related Work

Confidence metrics for classification

The confidence problem for image classification can be stated as follows: given an image X and a model h, output a prediction $h(X)$ along with a confidence score $CS(X)$ between 0 and 1, that correlates negatively with prediction uncertainty. Neural network classifier outputs (typically the $softmax$ function) can be seen as probabilities but are not a good confidence estimators.

There exist two families of solutions to the confidence problem:

- **Look at the neighbors**: a model is confident if its prediction for input X is similar to its predictions for neighbors of X.
- **Look at the shape of the output**: $softmax$ is a poor measure of confidence, but the statistics of $softmax$ outputs on a large enough sample gives us an indication of the confidence level.

There has been some reasonable success in providing confidence metrics for image classification using methods in these two families. We focus in this section on the neighborhood sampling family.

NS has been successfully used for image classification as done in (Jha et al. 2019) with attribution-based confidence (ABC). The core of the ABC algorithm is focused on improving performance by sampling mainly around pixels that have a high influence on the output.

However, at the time of writing and to the best of our knowledge, no work have been published with regards to confidence metric for object detection. The goal of this paper is to provide such metric.

Difference with classification

The confidence problem for object detection is a generalisation of the one for image classification, and can be stated as follow: given an image X and a model h, output several detected objects O_i, O_i being represented by a bounding box and a class, along with a confidence score $CS(O_i)$ that represents the confidence of the model in its prediction for object O_i.

In the context of object detection, the concept of confidence relates both to the position and class of detected objects (if there are any objects to detect to begin with), unlike a typical classification setting where uncertainty surrounds only the predicted class.

Note that we only consider confidence for detected objects. This means that if the model fails to detect an object altogether, it is not reflected in the returned score. Such extension to the definition of confidence might be the subject of future work.

Computational problem

Another concern is the computational problem. The essence of neighborhood sampling being to verify the predictions of the model in the neighborhood of the input, it is by construction very costly : in effect, if we sample N times, the computation time is increased by a factor of N. The goals of the methods for this family of solution has therefore been less to improve the reliability of the confidence metric, than to improve its performance by reducing the number of samples that need to be produced.

One such attempt is described in (Jha et al. 2019). The idea is that appropriate sampling suffers from the curse of dimensionality, and needs to be addressed in order to prevent an exponential increase in the number of samples that need to be produced. They address this by sampling around very targeted pixels, thereby dramatically reducing the number of samples that need to be produced to provide a good confidence score. This is done by calculating pixel attributions, i.e. pixels that contribute the most to the final prediction.

The problem with this method is that computing attributions for detection is an order of magnitude more complicated than for classification. This means two things:

1. Implementation is more tight to the architecture of the model used for object detection, partially removing the black-box feature.
2. The gain in performance we get by better targeting samples might be offset by the increase in complexity (and therefore computation) in the computation of attributions.

We investigated the second problem, and found that the use of attributions does not really increase performance. We describe these results in the appendix. Based on this, we use totally random sampling for our NS confidence metric.

NS Metric for Object Detection

Overview

The idea behind the NS confidence metric is to sample the neighborhood of the input, and measure conformance w.r.t. the original output.

The sampling of the input is performed the same way as for classification (Jha et al. 2019): by randomly turning off random pixels in the picture. Here, sampling is completely random, which is computationally efficient. As shown in

the appendix, experimental results indicate that performance gain using random sampling outweighs the gain with smart (but very slow) sampling, for instance by modifying preferentially high-attribution pixels.

The conformance w.r.t. the original output is a little more complex to compute than for classification. Here, conformance is computed at several levels:

- First, at image level, where we see whether the object from the original image is detected at all in the modified sample.
- Second, at bounding box level: here the original object has been detected in the modified sample, which means that bounding boxes have been drawn around the object. We then check whether (a) the class in the bounding box is the same as in the original bounding box, and (b) the number of overlapping bounding-boxes before *Non-Maximum Suppression* (NMS) (Rosenfeld and Thurston 1971) is roughly the same in the original image as in the modified sample, the term *roughly* being quantified as a specific hyperparameter to be tuned: if the percentage of difference between the number of overlapping bounding boxes is above a certain value, then the object is rejected as *non-conformal*.

Algorithm

In this section we describe the algorithm to compute the NS confidence metric.

Given an image X and a model h, we have a prediction that detects N objects $\{O_1, ..., O_n\}$. Each one of theses objects O_i is defined by the following:

- $BB(O_i)$: a bounding box around the object O_i
- $class(O_i)$: the class of the object O_i
- $n_{NMS}(O_i)$: the number of overlapping bounding boxes around O_i that have been suppressed by NMS.
- $P(O_i)$: the probability output for the class (usually the max of the output of the $softmax$ function for each class)

We want to compute an array $\{C_1, ..., C_n\}$ of confidence scores associated with each object.

The hyper-parameters of the NS confidence metric for detection :

- S: Size of the sample
- P: Percentage of pixels randomly turned-off.
- nms_τ: Percentage of difference between number of overlapping bounding boxes

We define the following function:
$\mathbf{Object}(\mathbf{h}, \mathbf{X_i}, \mathbf{O_j}) \rightarrow object$: detection of an object in a modified image

This function is made possible because *SSD*, like other implementations of detection using neural networks, has a predefined set of bounding boxes. We can therefore perform a 1-to-1 matching of detected objects in the bounding boxes for the original image and the modified sample.

$\mathbf{Conform}(\mathbf{O_1}, \mathbf{O_2}) \rightarrow boolean$: conformance of two objects

Algorithm 1: $\mathbf{Object}(\mathbf{h}, \mathbf{X_i}, \mathbf{O}) \rightarrow object$

Inputs:
h: Model
X_i: sample image
O: an object from the original image

Output: The object from X with the same bounding box as O

```
Let O_1, ..., O_N = h(X)
for i=1..N do
    if BB(O_i) == BB(O) then
        return O_i
    else
        return None
    end if
end for
```

Algorithm 2: $\mathbf{Conform}(\mathbf{O_1}, \mathbf{O_2}) \rightarrow boolean$

Inputs: two objects O_1 and O_2
Output: boolean: conformance of the two objects

```
if class(O_1) <> class(O_1) then
    return False
else
    if |(n_NMS(O_1) - n_NMS(O_2)) / n_NMS(O_1)| > nms_τ then
        return False
    else
        return True
    end if
end if
```

Features of the NS confidence metric

The features are the following:

It is black box. By this, we mean that the algorithm requires no knowledge of (a) the artitecture of the model, and (b) the weight of the neural network. For that matter, the model could something completely different from a deep neural network, the method would still work.

It is easy to implement. Although the method is simple and black-box, we make for convenience some assumptions about the structure of the model. It is important to note that these assumptions can be relaxed.

- The model is a deep neural network, that outputs a large set of bounding boxes from the original input, along with a class and a probability for this class. This assumption is in ligne with most state-of-the-art detection models
- The output above goes through a non-maximum suppression (NMS) phase, that merges overlapping bounding-boxes around a same object. This phase is put in use to fine-tune the confidence score, but is not necessary.

The performance is easy to calculate. Given the algorithm described above, execution time is A + St, where

- t is the execution time of the model
- S is the number of samples

Algorithm 3: calculate-confidence(I, M) : Float

Inputs: Model M, image I
Output: Confidence score

Let $O_1, ..., O_N = M(I)$
Let $C_1, ..., C_N = 1, ..., 1$
Let $I_1, ..., I_S$ = S samples of I (generated by randomly turning off P pixels of I)
for i=1..S **do**
  for j=1..N **do**
    if $object(I_i, O_j) is None) or (notConform(O_j, object(I_i, C$
    then
      $C_j := C_j - 1/S$
    end if
  end for
end for
return solution

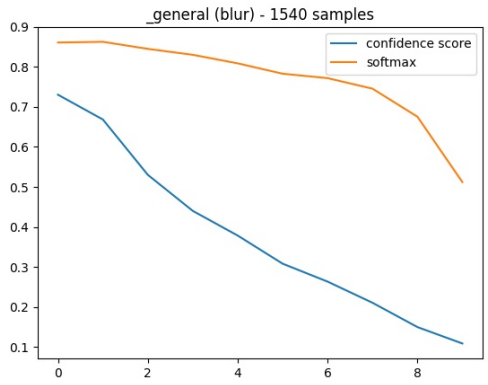

Figure 3: SoftMax vs NS confidence metric on all monoclass instances from the Coco2017 validation dataset.

- A is a constant

The number of samples is therefore a key parameter to assess the performance of the NS confidence metric. The evaluation of a proper number of samples to use is discussed in the next section.

Experimental Evaluation

Overview

Goal We performed various evaluations of the NS confidence metric. The goal of the evaluation is threefold: Firstly, to validate the method. In other words, make sure that the metric gives a good assessment of the confidence the model has in its prediction. A more precise definition of such confidence is given in the next paragraph. Secondly, to evaluate the performance of the metric. Practically, since the performance is closely linked to the number of samples, we experimentally evaluate the number of samples required to produce good results on a given dataset. Lastly, to evaluate the stability of the hyperparameters across datasets, and across classes within a dataset. In other words, evaluate experimentally the necessity to calibrate the metric.

Implementation In order to test the validity of the score produced by the NS confidence metric, we feed the model with some data we know will give some poor prediction, hopefully with a capacity to calibrate.

In other words, given an image I from the validation dataset, we want to product an image I' = modif(I, p) where p is a parameter between 0 and 9, and modif(I, p) is a modified version of I with a modification factor of p : I' is equal to I when p=0, and I' is almost completely unrecognizable when p=9.

What we expect then, is to have the confidence score decrease when p increases, and to quantify this evolution.

Use-cases For our experiments, we used the gaussian blur as the modifications method : we blur our image with a blurring factor p ranging from 0 to 9. At $p = 0$, the image is not blurred at all, and we expect a good detection of objects (the dataset being a validation dataset, the confidence score is not expected to be close to 100%). At p=9, the image is almost unrecognizable even for a human, and we expect the confidence score to be close to zero.

We tested this method against the COCO 2017 dataset. From the validation dataset of coco, we extracted all the images that contained only one object. This produced a dataset of 1540 images. Although our method works well on multiple objects, doing so was more convenient for charting results.

Hyper-parameters The second and third goals of our experiments are closely linked to the hyper-parameters. The three main hyper-parameters we want to tune are the following:

- **Number of samples (N_S).** This hyper-parameter is the most important one with regards to performance. Our goal is therefore to lower their required number of samples as much as possible, and possibly tune the other hyper-parameters to get good results even with a low number of samples.
- **Probability for an feature to be turned off (p).** In other words, this hyper-parameter is how much "salt" we add to produce a sample in the neighborhood of the input.
- **NMS threshold (nms_τ).** As described in the algorithm for the score calculation, we discard a detected object in the sample when the number of overlapping bounding boxes that have been suppressed by the NMS differs in the sample than in the original input. This hyper-parameters is the threshold for the discard condition. If nms_τ is set to .9, this means that the number of overlapping bounding boxes have to differ more than 90% for the object in the sample image to be discarded as non matching.

Results

Figure 3 as been produced with increasingly blurring the input images, and with the following hyperparameters for calculation the confidence score: $nb_samples = 100$; $p = .0$;

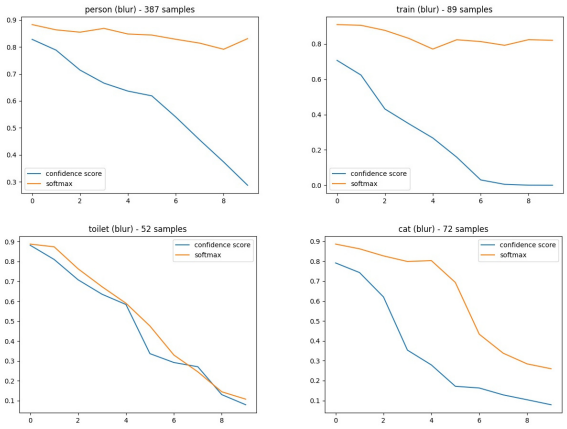

Figure 4: The pertinence of the NS confidence score as compared to the softmax output varies depending on the class. Top: person; pizza. Bottom: toilet ; cat

$nms_th = .9$. The graph is a mean value of all the scores for all of the images in the dataset. As we see, the softmax output is overconfident, while the NS confidence score decreases linearly with the blur factor.

As expected, the softmax output gives overconfident predictions even when the blur factor is high. On the other hand, the confidence score drops progressively with the blur factor.

When we look at the results for specific class, we observe some variations of the results depending on the class. Certain classes (person, pizza) present exceptionnaly good results for the confidence score as compared to the softmax output. Other classes have a good confidence metric, but the softmax output also presents good confidence results. Still other classes have been discarded as not enough samples are present.

Conclusion

We use neighborhood sampling to provide a confidence metric on image detection. This method gives good results for classification, and we applied the same principle to detection with reasonable success. Our method uses vanilla neighborhood sampling, and we validated experimentally that this works with reasonable performances (and sample of size 50 is sufficient in our settings).

So far, our experimentation is based on one dataset and one method of image modification, whose goal is to make the prediction less accurate and to experimentally verify a drop in the confidence score. As such, the obvious next steps are (1) to use other datasets, and (2) to use other image modification methods. For (2), we identified the following: (a) use poisson-blending with images that are completely out of the dataset; (b) use adversarial attacks.

Also, the metric can potentially be improved by including in the calculation the bounding boxes around the object, the detect another class than the true class, but that are discarded by the NMS.

Finally, despite the results on classification, it might be interesting to test the use of attributions to speed up calculation in the case of image detection.

Acknowledgments

This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute.

Appendix

Neighborhood sampling has been successfully used to calculate confidence in image classification in (Jha et al. 2019), but with the caveat that for images the neighborhood is high-dimensional, leading to a computational challenge that is solved by lowering the dimensions around high-attribution features.

We implemented the code for (Jha et al. 2019) and reproduced the good results for the following datasets: MNIST, FashionMNIST, and Cifar10. But we also tested these results when the high-attribution calculation of the code is removed.

The method is to perform predictions on the validation data of the aforementioned datasets, as well as on transformed data. The goal of the transformation is to force the model to produce invalid predictions. The transformations are (a) rotation (parameterized by the angle of rotation), and (b) alpha-blending with a random image in the dataset (parameterized with the percentage of blending).

The results in figure 5 show the average confidence score for all predictions in the validation dataset for the various transformations. We show the confidence scores calculated with and without focusing on high-attribution features only. Most importantly, these results are calculated using the exact same number of samples, which means that the calculation without using attributions is actually faster, because it does not include the cost of calculating the attributions for the input.

We conclude from this experiments that the use of attributions does not increase in practice the performance of confidence score calculation using neighborhood sampling, and it is reasonably safe to remove this additional computation when performing neighborhood sampling.

References

Jha, S.; Raj, S.; Fernandes, S.; Jha, S. K.; Jha, S.; Jalaian, B.; Verma, G.; and Swami, A. 2019. Attribution-Based Confidence Metric For Deep Neural Networks. In *NeurIPS Proceedings*.

Lam, D.; Kuzma, R.; McGee, K.; Dooley, S.; Laielli, M.; Klaric, M.; Bulatov, Y.; and McCord, B. 2018. xview: Objects in context in overhead imagery. *arXiv preprint arXiv:1802.07856*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*, 21–37. Springer.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Ren, K.; Zheng, T.; Qin, Z.; and Liud, X. 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering*, 6(3): 346–360.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. 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.

Rosenfeld, A.; and Thurston, M. 1971. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, 100(5): 562–569.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

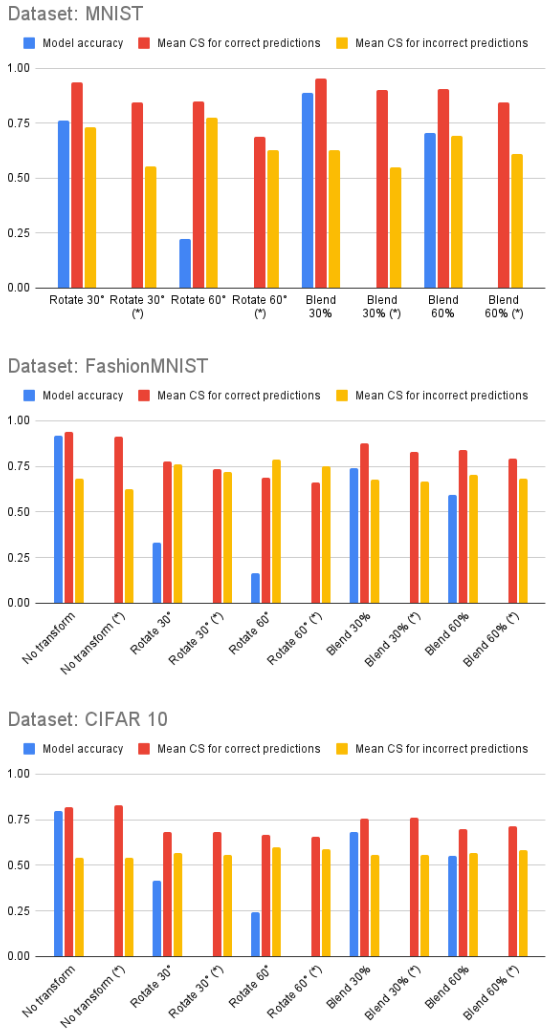

Figure 5: Confidence score with (no asterisk) and without (asterisk) the use of attribution-based sampling. Transformations are used to force the model to produce erroneous predictions. The confidence score (CS) for both correct and incorrect prediction is calculated. We see that the mean CS is very similar across datasets, and for different transformations, whether we use attribution-based sampling or not.