Stochastic-YOLO: Efficient Probabilistic Object Detection under Dataset Shifts

Tiago Azevedo[*]
Department of Computer Science and Technology
University of Cambridge
tiago.azevedo@cst.cam.ac.uk

René de Jong
Arm ML Research Lab
rene.dejong@arm.com

Matthew Mattina
Arm ML Research Lab
matthew.mattina@arm.com

Partha Maji
Arm ML Research Lab
partha.maji@arm.com

Abstract

The robustness of object detection (OD) tasks under a probabilistic framework have become a question of paramount interest in autonomous driving applications. However, unlike in image classification, OD tasks pose other challenges for uncertainty estimation and evaluation. For example, one needs to evaluate both the quality of the label uncertainty (i.e., *what?*) and spatial uncertainty (i.e., *where?*) for a given bounding box, but that evaluation cannot be performed with more traditional average precision metrics (e.g., mAP). In this paper, we adapt the well-established YOLOv3 architecture to generate uncertainty estimations by introducing stochasticity in the form of Monte Carlo Dropout (MC-Drop), and evaluate it across different levels of dataset shift. We call this novel architecture Stochastic-YOLO, and provide an efficient implementation to effectively reduce the burden of the MC-Drop sampling mechanism at inference time. Finally, we provide some sensitivity analyses, while arguing that Stochastic-YOLO is a sound approach that improves different components of uncertainty estimations, in particular spatial uncertainties.

1 Introduction

Current developments in computer vision for autonomous driving (AD) applications are sustained by very powerful deep learning models, with proven capabilities to deliver good results. Due to its direct impact in daily human lives, it is of paramount importance to have models that generalise well to unseen cases during training time.

This is well explored for image classification tasks, where it has been seen that many models get wrong very confidently as dataset shift increases [13]. But object detection (OD) tasks pose other types of challenges, as the metrics used in image classification tasks can be quite distinct. For example, in OD tasks, one has none to many bounding boxes being predicted for just a single image, in which *uncertainty* semantics have a more diverse meaning. While in both image classification and OD tasks we can measure how certain the model is about *what* the labels might be, in the OD case we also need to know how certain the model is about *where* the objects (i.e., bounding boxes) are located. These two measures of uncertainty – related to label and spatial information – can be evaluated using a recently introduced metric named Probability-based Detection Quality (PDQ) [3] (see Section 3.3).

---

[*]Work performed while the author was at Arm ML Research Lab

We introduce Stochastic-YOLO, a novel architecture adapted from the well-established YOLOv3 [15], into which we add Monte Carlo Dropout (MC-Drop) sampling to introduce stochasticity in the predictions, which in turn generate uncertainty estimations. Although there are alternatives to introduce stochasticity, MC-Drop is computationally light-weight and scales well at inference time. Ideally, a Bayesian neural network [17, 1] would provide a fully probabilistic framework with more precise measures of uncertainty, but Bayesian nets have the downside of significant memory footprint. Likewise, an ensemble of models is known in the literature to produce better class labels and uncertainty predictions [13, 16], but it also brings an obvious memory footprint and expensive training times. We claim that using MC-Drop in OD tasks is the best trade-off between cost and robustness to dataset shifts under a probabilistic framework.

Similarly to previous works on image classification tasks, we will systematically evaluate how OD models generalise on increasing levels of corruptions, mainly in terms of label and spatial uncertainties. Although all these topics are somehow scattered throughout literature, to the best of our knowledge, no recent paper systematically evaluated label and spatial quality of OD techniques in increasing levels of dataset shifts in a single work, with inference time performance in mind. We also provide some sensitivity analysis on specific decisions for MC-Drop implementation which we did not find in the literature. Our contributions can be summarised in the following three main points: (1) improvements on YOLOv3 architecture with an introduction of MC-Drop and sensitivity analysis over important hyperparameters, (2) introduction of an efficient caching mechanism for MC-Drop, effectively reducing sampling burden at inference time, which can be directly adapted to other models, and (3) leveraging probabilistic-based metrics (e.g., PDQ) to systematically evaluate model robustness for different levels of uncertainty (i.e. spatial and label qualities), while showing MC-Drop is a sound approach to improve PDQ.

The code used in this work is publicly available in the following Github link: `https://github.com/tjiagoM/stochastic-YOLO/`.

2   Related Work

There are many attempts in the literature to bring stochasticity to OD models in order to capture their prediction confidence. Such attempts can be summarised in four categories: (1) directly learning to output Gaussian parameters for each bounding box coordinate, (2) using Bayesian approaches (e.g., Bayesian neural networks) to have a full probabilistic model, (3) using sampled-based Bayesian approximations like Monte Carlo Dropout (MC-Drop), and (4) using ensemble of models that generate a distribution of predictions, which can also be approximated as Gaussian parameters.

Uncertainty in Object Detection (OD):   Gaussian YOLOv3 [2] adapted the YOLOv3 architecture [15] and corresponding loss so the model could output Gaussian parameters instead of single, deterministic coordinates. This approach significantly reduced the false positive cases while keeping similar inference time when compared to YOLOv3. Similarly, He et al. [4] proposed a new loss named KL Loss to learn localisation uncertainty (i.e., variance) together with bounding boxes, which could then empower a voting scheme to select bounding boxes. Another work [5] on a large scale automotive pedestrian dataset also reimplemented YOLOv3 in order to estimate epistemic and aleatoric uncertainty by using MC-Drop. Finally, another related work [12] used the variance introduced by MC-Drop on a YOLOv3 architecture to measure spatial uncertainty; as a result, detections of lunar craters could be accepted or rejected based on that variance. However, the spatial quality of those bounding boxes were not directly evaluated using specific quantitative metrics and these detections did not need to be performed in real-time like Stochastic-YOLO is targeted for.

Robustness to non closed-set data: Ovadia et al. [13] claim to have conducted the first rigorous, large-scale empirical comparison of predictive uncertainty for probabilistic deep learning methods under dataset shift. However, this study was conducted in an image classification task, leaving it open how these methods would behave in OD tasks. Indeed, some recent works analysed these issues in more detail for OD tasks. One work [10] claims to be the first to investigate the utility of dropout sampling for OD. The authors were able to show that dropout variational inference is able to improve OD performance under open-set conditions when compared to non-bayesian models, even though they did not evaluated how good the spatial uncertainty was for their open-set datasets. Posterior works further extended this analysis: one work [10] compared different combinations of

affinity measures and clustering methods for MC-Drop with a Single Shot Multi-Box Detector (SSD) in closed set data, near open-set data, and distant open-set data. Furthermore, some authors [11] also analysed the performance of MC-Drop and Deep Ensembles in one- and two-stage OD models, showing Deep Ensembles yielding the best results. They have also introduced a new merging strategy that achieved comparative performance with only one hyperparameter needed, compared to three from other models, which we used in this paper (see Section 3). Robustness to OOD cases can also be seen under the framework of adversarial attacks [18]; however, we willingly did not tackle that research direction in this work and focused instead on more systematic dataset shifts.

All these works showed advantages in considering localisation/spatial uncertainty in OD models for non closed-set data. Spatial uncertainty is many times used to help in deciding which bounding boxes are correct, but there was not an unified way to systematically evaluate measures of uncertainty for different open-set scenarios. Indeed, from the OD models presented in this section, only one [11] used the unifying metric for probabilistic OD that we use in this paper (i.e., PDQ). Our work also has the objective of introducing stochasticity for good predictive uncertainty measures in a way that can be efficiently run at inference time, which we do not see in previous related literature.

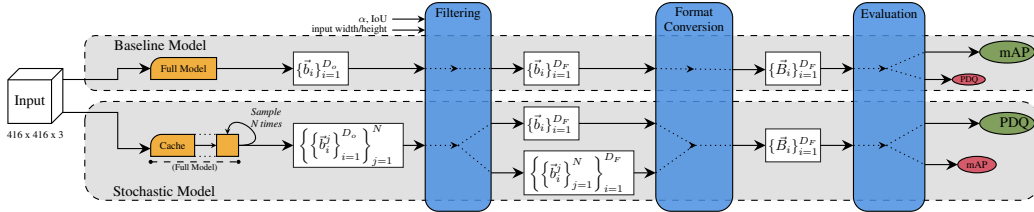# 3 Methods

3.1 Stochastic-YOLO

Figure 1: Working blocks of a deterministic baseline model and a stochastic model with Monte Carlo Dropout, from which YOLOv3 and Stochastic-YOLO are special cases. A stochastic model outputs better probabilistic scores (i.e., PDQ) when compared to a deterministic baseline that outputs inflated mAP metrics. These inflated mAP metrics give a false sense of model performance.

We introduce Stochastic-YOLO, a novel OD architecture based on YOLOv3 [15] with efficiency in mind. We added dropout layers for Monte Carlo Dropout (MC-Drop) sampling in each one of the three YOLO heads; in specific, each head had a dropout layer before the last CNN layer, and before the second to last CNN layer. We argue that MC-Drop brings a good trade-off in terms of complexity and efficiency for introducing stochasticity at inference time.

The introduction of dropout layers towards the end of a deep OD model, instead of after every layer, allows for an efficient sampling procedure. When sampling N times from a deep OD model, one can cache the intermediate resulting feature tensor of one forward pass right until the first dropout layer. This cached tensor is deterministic (assuming that numerical errors are not significant), thus allowing only the last few layers of the model to be sampled from instead of making N full forward passes.

For the specific case of Stochastic-YOLO, MC-Drop is applied at inference time by sampling 10 times from the model with the dropout layers activated. The suppressions included in this work, both for Stochastic-YOLO and YOLOv3, consisted of (1) removing bounding boxes below a certain confidence threshold α (we tested for 0.1 and 0.5), (2) removing bounding boxes in which at least one coordinate was located outside image margins, and (3) removing bounding boxes with an intersection over union (IoU) above 0.6 with another bounding box containing a higher classification score. These suppressions are calculated on the coordinates averaged from the 10 samples, thus closely following the Pre-NMS technique introduced by Miller et al.'s [11]. There are other techniques in the literature for suppressing bounding boxes when using MC-Drop, but this one revealed to have a good trade-off between speed and performance when compared to other bounding box merging techniques [11].

This entire process is summarised on a higher-level for non-stochastic and stochastic cases in Figure 1, from which YOLOv3 and Stochastic-YOLO are special cases. For the non-stochastic baseline model at the top, its output is a set $\{\vec{b}_i\}_{i=1}^{D_o}$, where $\vec{b}_i \in \mathbb{R}^{5+C}$ is a bounding box predicting C possible labels,

and D_o is the number of bounding boxes originally produced by the specific model (e.g., YOLOv3). Each bounding box contains $5 + C$ real values: four representing the bounding box (i.e., x and y coordinates, and width and height), one representing the objectness score (i.e., softmax score that the bounding box contains an object), and C softmaxed scores for each possible label. This set enters the *Filtering* block which contains the suppression techniques described in the previous paragraph, now producing a smaller set $\{\vec{b}_i\}_{i=1}^{D_F}$ of bounding boxes, where D_F is the final number of bounding boxes to be evaluated. For the stochastic model, the number of bounding boxes being originally produced is instead $N \times D_o$, where N is the number of MC-Drop samples. This distinction in the stochastic model's output, when compared to the non-stochastic one, will make the *Filtering* block to have an extra output: for each averaged bounding box $\vec{b}_i$ not being filtered, we need the corresponding N samples of that bounding box, represented as $\left\{\vec{b}_i^j\right\}_{j=1}^{N}$.

A further *Format Conversion* block is needed to transform these sets into a format which can be evaluated from a probabilistic perspective in the *Evaluation* block (see Section 3.3). In practice, a bounding box vector $\vec{b}_i = [x, y, width, height, \mathrm{obj}, p_0, ..., p_C]$ is transformed into $\vec{B}_i = [\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, \Sigma^1, \Sigma^2, \hat{p}_0, ..., \hat{p}_C]$, where now we have two coordinates for the top-left corner (i.e., $(\bar{x}_1, \bar{y}_1)$) and bottom-right corner (i.e., $(\bar{x}_2, \bar{y}_2)$) instead of a single coordinate with width/height. In this work, for a deterministic model these values correspond to those originally in $\vec{b}_i$, whereas for the stochastic model these are the average coordinates across the N samples. Each softmaxed score is transformed such that $\hat{p}_i = p_i \times \mathrm{obj}$. Finally, Σ^i is a covariance matrix for one coordinate with the following representation:

$$\Sigma^i = \begin{bmatrix} \Sigma_{xx}^i & \Sigma_{xy}^i \\ \Sigma_{yx}^i & \Sigma_{yy}^i \end{bmatrix},$$ (1)

in which for the deterministic model these will all be zeros, whereas for the stochastic model each covariance matrix is calculated from the distribution of the N sampled points in each coordinate. When a covariance matrix is not positive semi-definite, we transform it by computing the eigen decomposition, and reconstructing the matrix again with eigenvalues of zero where they were previously negative.

We highlight that the reasoning in this section illustrated in Figure 1 can be applied in almost any OD model that outputs some set of bounding boxes, even though in our experiments we illustrate this for YOLOv3 and Stochastic-YOLO only. Indeed, researchers might want to consider other merging strategies for their specific needs [9].

3.2 Systematic Evaluation of Dataset Shift Scenarios

We used the Python package proposed by Michaelis et al. [8] to systematically evaluate the robustness of our models to increasing dataset shifts. Among other contributions, the package provides a total of 5 severity levels across 15 corruption types to be applied in images, some of them corresponding to distortions that can be seen more often in real-world scenarios, like rain, snow, and fog. From these 15 corruptions, we used all but glass blur due to the extended time taken to be calculated, when compared with all the others.

Michaelis et al. [8] also presented a single measure to evaluate relative performance degradation under corruption, named relative performance under corruption (rPC):

$$\mathrm{rPC} = \frac{\frac{1}{N_c} \sum_{c=1}^{N_c} \frac{1}{N_s} \sum_{s=1}^{N_s} P_{c,s}}{P_{\mathrm{clean}}},$$ (2)

where $P_{c,s}$ is the performance of the model for severity level s on corruption c, N_c is the number of corruptions applied (i.e., 14 in our case) and N_s the number of severity levels (i.e., 5 in our case). P_{clean} corresponds to the performance of the model without any corruption in the dataset. Note that in the original paper the authors only used mAP as a performance metric, but this can be applied in any performance metric. For instance, rPC_{m_i} corresponds to the relative performance of the model under corruption for metric m_i.

3.3 Probability-based Detection Quality (PDQ)

We use the Probability-based Detection Quality (PDQ) metric [3], which is calculated between the set of ground truths G, and the set of detections D. This metric is built on top of two ideas: label quality and spatial quality.

Label quality Q_L on the f-th image (named *frame* instead of *image* in the original paper) between a i-th ground-truth object G_i^f and a j-th detection D_j^f, can be defined as:

$$Q_L\left(G_i^f, D_j^f\right) = \mathbf{I}_j^f\left(\hat{c}_i^f\right),\tag{3}$$

where $\mathbf{I}_j^f$ is the probability distribution across all classes on the j-th detection object, and $\hat{c}_i^f$ the ground-truth class. Note that this ignores whether the class is the highest ranked in the probability distribution $\mathbf{I}_j^f$, thus effectively evaluating the quality of *what* object is being predicted.

Spatial quality Q_S on the f-th image between a i-th ground-truth object G_i^f and a j-th detection D_j^f, can be defined as:

$$Q_S\left(G_i^f, D_j^f\right) = \exp\left(-\left(L_{FG}\left(G_i^f, D_j^f\right) + L_{BG}\left(G_i^f, D_j^f\right)\right)\right),\tag{4}$$

where L_{FG} and L_{BG} are two loss terms for the foreground and background of that image, respectively. This spatial quality will be equal to 1 (i.e., maximum value) when the detector D_j^f assigns a probability of 1 to all the ground-truth pixels in G_i^f, while not assigning any probability to the other pixels (i.e., background). The probability distribution used to measure these two losses is calculated by joining the two covariance matrices from each corner introduced in Equation 1. For details on how to derive the final PDQ score, we refer the reader to the original paper [3].

4 Experiments and Results

4.1 Training Procedure

We have used and adapted Ultralytics' open-source implementation[2] of YOLOv3 for Pytorch [14] with input size of 416×416. Training followed all the repository's default hyperparameters, unless stated otherwise. Baseline YOLOv3 model was trained for 200 epochs using Stochastic Gradient Descent, cosine learning rate decay [7], and data augmentation. Final model is selected according to best mAP in the validation set. Training and evaluation was performed using the *train2017* and *val2017* splits provided by MS COCO [6] 2017 release, with 118,287 and 5,000 images, respectively. Stochastic-YOLO is created by inserting the dropout layers in the best YOLOv3 model as previously described. For comparison, we train an ensemble of five YOLOv3 models, where each one was trained in the same way with different random seeds when initialising the network's weights.

YOLOv3 baseline model was trained with a batch size of 20, distributed across 2 *GeForce RTX 2080* GPUs, producing training times of around 48 minutes per epoch. Ensemble models were trained with a batch size of 40, distributed across 2 *GeForce RTX 2080 Ti* GPUs, producing training times of around 35 minutes per epoch. When Stochastic-YOLO is fine-tuned, this is done under the same conditions as YOLOv3, but with dropout layers activated at training time and for 50 extra epochs.

4.2 Robustness to Corruptions

Table 1 summarises metrics and their relative performance under corruption (rPC) across different models. For every model we show two confidence thresholds – 0.1 and 0.5 – and for all the models a confidence threshold of 0.1 corresponds to higher values of mAP and better robustness for mAP. However, for the label and spatial quality metrics (and corresponding relative performances under corruption), all the models perform significantly worse when a confidence threshold of 0.1 is used. This illustrates well-known concerns in the field, where many developers try to decrease the confidence threshold to inflate mAP values, giving a false sense of good performance.

---

[2]https://github.com/ultralytics/yolov3

Table 1: Overall results across different models and metrics, where Lbl and Sp mean average label and spatial uncertainty quality, respectively. In parenthesis confidence threshold. S-YOLO means Stochastic-YOLO in which the corresponding number is the dropout percentage applied, and -X means fine-tuned model. In bold best results for each metric.

	Deterministic Metrics (%)		Probabilistic Metrics (%)					
	mAP	rPC$_{mAP}$	PDQ	rPC$_{PDQ}$	Lbl	rPC$_{Lbl}$	Sp	rPC$_{Sp}$
YOLOv3 (0.1)	34.43	20.64	7.19	4.88	55.27	48.38	14.53	12.18
YOLOv3 (0.5)	26.07	14.64	9.26	5.40	**72.88**	**69.70**	18.42	16.11
Ensemble-5 (0.1)	**37.14**	**22.28**	18.36	**13.38**	50.66	43.25	39.24	37.66
Ensemble-5 (0.5)	26.00	14.07	19.53	11.50	72.21	**69.63**	49.21	48.59
S-YOLO-25 (0.1)	31.67	19.31	17.73	**13.40**	48.18	41.67	37.27	34.84
S-YOLO-25 (0.5)	22.72	12.91	**20.27**	12.46	70.17	67.70	45.78	44.95
S-YOLO-25-X (0.1)	33.16	19.66	17.29	11.95	51.01	44.62	38.59	34.58
S-YOLO-25-X (0.5)	24.20	13.58	19.08	11.39	70.99	68.09	48.76	45.58
S-YOLO-75 (0.1)	17.36	11.11	4.44	4.20	29.88	24.79	19.71	17.79
S-YOLO-75 (0.5)	7.47	4.36	7.81	5.09	61.32	60.71	25.88	26.61
S-YOLO-75-X (0.1)	29.36	17.63	17.86	12.57	42.94	38.68	39.36	36.82
S-YOLO-75-X (0.5)	17.76	10.11	15.20	9.51	66.34	64.41	**52.27**	**49.97**

Fine-tuning Stochastic-YOLO models usually brings better metrics and robustness to corruptions when compared to Stochastic-YOLO used directly from a pre-trained YOLOv3 model with inserted dropout layers and no fine-tuning. Although for a dropout rate of 25% these are just slight improvements, for a dropout rate of 75% they are much more substantial, even resulting in the best spatial quality and robustness for spatial quality among all models.

PDQ score, spatial quality and robustness for these two metrics more than doubled for Stochastic-YOLO model with 25% dropout rate when compared to YOLOv3. At the same time, label quality, robustness for label quality, and robustness for mAP only reduced around 2% when compared to YOLOv3 with the same 0.5 confidence threshold. The mAP metric is the only one that is more significantly impacted (almost 4%) with Stochastic-YOLO for a confidence threshold of 0.5 at 25% dropout rate, but the issues with mAP usefulness are well-known in the field.

It is noteworthy the very good label quality across all models when a confidence threshold of 0.5 is used. The disproportionate capability of YOLOv3 to produce very high label quality measures but very low spatial quality measures has been previously seen in the literature [3]. This is also the performance metric more robust to corruptions, with a relative performance (i.e., rPC$_{Lbl}$) of almost 70% in the best models. By using a confidence threshold of 0.1 instead of 0.5, these two metrics are the most severely impacted.

4.3 Model Fine-tuning

As it can be seen in Table 1, the decision of further fine-tuning Stochastic-YOLO can impact overall results. We check three representative dropout rates (25%, 50%, and 75%) and resulting spatial and label qualities over corruptions averaged across dataset shifts. These results are depicted in Figure 2 for label and spatial quality, with a confidence threshold of 0.5.

Both uncertainty qualities (i.e., label and spatial) follow similar patterns: fine-tuning Stochastic-YOLO produces better averaged metrics across increased levels of dataset shifts, with these improvements being smaller for a smaller dropout rate (i.e., 25%), and more noticeable for a higher dropout rate (i.e., 75%). For spatial quality this behaviour is even more evident, as the best averaged spatial qualities are produced when fine-tuning with 75% dropout rate, followed by 50% and 25%. All these three fine-tuned models are the best, on average, for most cases when compared to all the other non fine-tuned models.

Nevertheless, Stochastic-YOLO with a dropout rate of 25% with no fine-tuning seems to yield the best trade-off in terms of performance and complexity, as it achieves comparable results without the need to spend further time fine-tuning it.
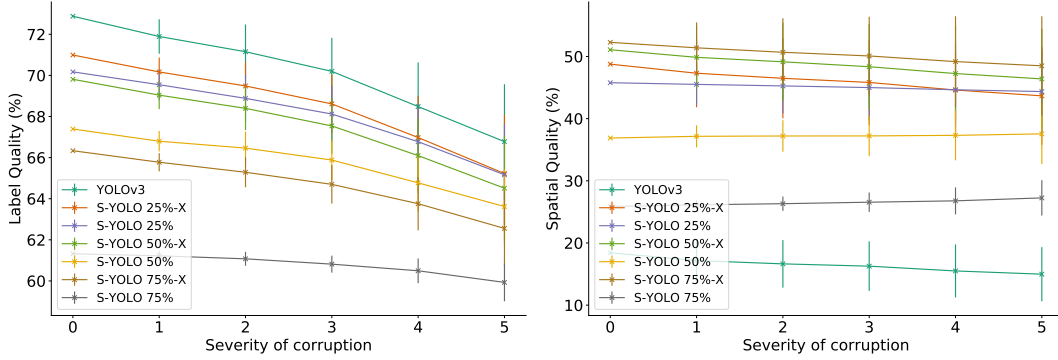
Figure 2: Label Quality and Spatial Quality averaged across all corruptions for each severity, with 0.5 confidence threshold. Error bars correspond to one standard deviation. S-YOLO means Stochastic-YOLO followed by dropout rate. -X means fine-tuned model. Note the reduced variation in the y-axis for the Label Quality.
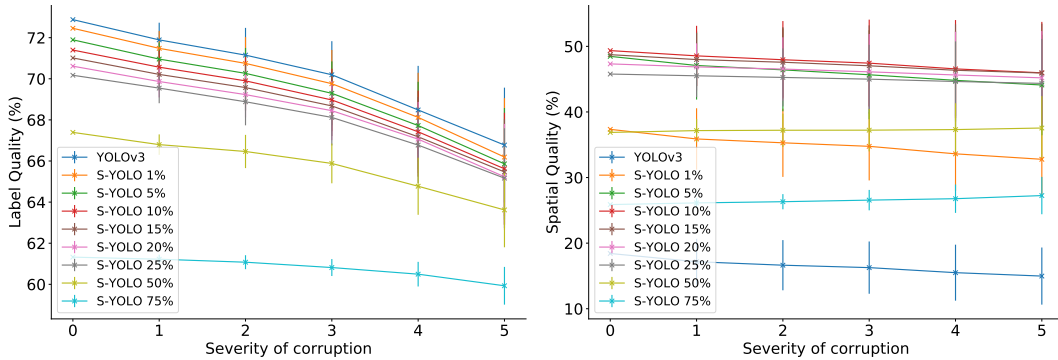

Figure 3: Label Quality and Spatial Quality across different dropout rates, averaged across all corruptions for each severity, with 0.5 confidence threshold. Error bars correspond to one standard deviation. S-YOLO means Stochastic-YOLO. Note the reduced variation in the y-axis for the Label Quality.

4.4 Sensitive Analysis on Dropout Rate

We provide a sensitivity analysis on the influence of dropout rate, a hyperparameter that needs to be defined when using Stochastic-YOLO. Due to resource and time constrains, we focus this sensitivity analysis on Stochastic-YOLO without any further fine-tuning. Figure 3 contains the results for label and spatial quality, with a confidence threshold of 0.5.

The behaviour of Stochastic-YOLO is very consistent for label quality across increased dataset shifts: the smallest dropout rate produces the best label quality (only surpassed by YOLOv3), and consistently worsens for higher dropout rates. However, for the spatial quality the behaviour is not the same. There is a range of very similar results between 5% and 25%, with significant worse results below and above this range. Indeed, on average, with 1% dropout rate Stochastic-YOLO yields worse results when compared to 50% for spatial quality. Once again, Stochastic-YOLO with 25% dropout achieves a good trade-off across different metrics without fine-tuning.

4.5 Model Efficiency

We measure the impact of the networks on inference time and power usage, by setting up an experiment running the different models on two different real devices. For this, we make use of the NVIDIA Jetson TX2 (TX2) platform and the NVIDIA GeForce RTX 2080 (RTX 2080) graphics card. Both platforms allow for an accurate power measurement during runtime. The TX2 has a Dual core Denver 2 CPU (2GHz) and a Quad-Core Arm Cortex-A57 CPU (2GHz). It comes with a 256-core NVIDIA Pascal GPU, and 8GB LPDDR4 memory. The RTX 2080 is a Turing architecture GPU with 2944 CUDA cores and 8GB GDDR6 memory. Each network inference is run 1000 times and
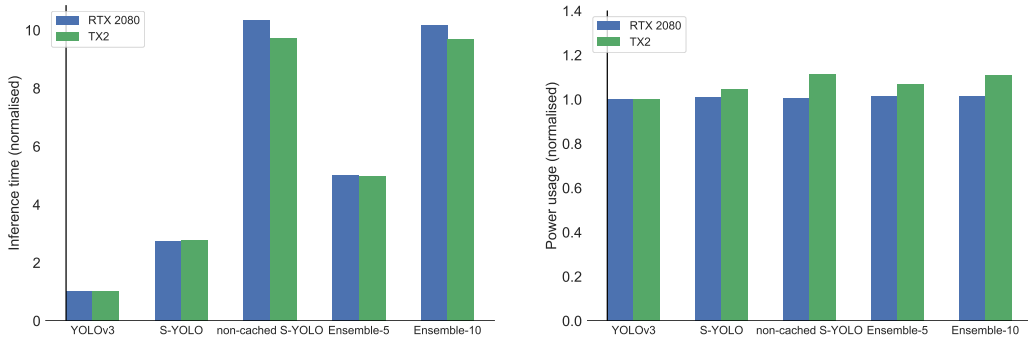
Figure 4: Comparison of inference time (left) and power usage (right) of the different models normalised to the inference time (left) and power usage (right) of YOLOv3 on that platform, as run on a NVIDIA RTX 2080, and a NVIDIA Jetson TX2.

throughout the run we measure the power. At the end of each experiment we calculate the average power consumption and inference times of each network.

By performing measurements on both platforms, we illustrate the effect of the different model performances for high-performance systems as well as energy-efficient systems. The results of these measurements are summarised in Figure 4. All the results are normalised to the results as measured for the YOLOv3 run on that platform. The measured inference time for YOLOv3 is 12 ms per instance on the RTX 2080, and 162 ms on the TX2. Its power usage is 216 W and 9 W for the RTX 2080 and the TX2, respectively. "Ensemble-10" was created only for comparison purposes, and corresponds in practice to duplicating the models from "Ensemble-5".

We see that the baseline run of YOLOv3 shows in all cases the lowest inference times, and the non-cached Stochastic-YOLO and the Ensemble show the highest. Since this non-cached version of Stochastic-YOLO is run ten times, we see an unsurprising ten-time slowdown in the inference times. However, the overhead for the cached model only shows a two to three times slowdown compared to the baseline and a three to four times speedup compared with the non-cached Stochastic-YOLO. The power consumption does increase only marginally in comparison with YOLOv3. This demonstrates that Stochastic-YOLO's caching mechanism is an efficient alternative to the full Stochastic-YOLO without any caching mechanism.

5 Discussion

In this work, we enhanced the well-established YOLOv3 architecture in order to generate uncertainty estimations in OD tasks. These improvements included the addition of Monte Carlo Dropout (MC-Drop), thus successfully generating both label and spatial uncertainties. The implementation of this architecture was achieved using a caching mechanism that enabled minimal impact on inference time when deploying the model in the real world. All these developments allowed for better resulting probabilistic metrics (i.e., PDQ), which have proven advantages over the more traditional mAP metric that can sometimes give a false sense of performance.

Although in the literature the use of MC-Drop usually includes dropout layers activated at both training and test times, in many OD architectures dropout is not a typical choice of regulariser for training. As a consequence, we developed Stochastic-YOLO in a way that can fit most of OD researchers current pipelines, in the sense that we show that direct application of MC-Drop in pre-trained models without any fine-tuning results in significant improvements (see Table 1). To further help the community decide how to use Stochastic-YOLO, we provided sensitivity analysis on dropout rate and on the decision of further fine-tuning the model, which to the best of our knowledge has not yet been fully explored in the field. Overall, we found that not fine-tuning Stochastic-YOLO yields sufficiently good results already, in which case a smaller dropout rate should be picked. For those with the time and computational resources for further fine-tuning their stochastic models with MC-Drop, a higher dropout rate should be picked instead. We hope that this work encourages other researchers to extend the ideas presented in this paper on their own models (as illustrated in Figure 1) with the help of our publicly available code.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

[2] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[3] David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Sunderhauf. Probabilistic object detection: Definition and evaluation. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020.

[4] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019.

[5] Florian Kraus and Klaus Dietmayer. Uncertainty estimation in one-stage object detection. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, Oct. 2019.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer International Publishing, 2014.

[7] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2016.

[8] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[9] Dimity Miller, Feras Dayoub, Michael Milford, and Niko Sunderhauf. Evaluating merging strategies for sampling-based uncertainty techniques in object detection. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019.

[10] Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sunderhauf. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018.

[11] Dimity Miller, Niko Sunderhauf, Haoyang Zhang, David Hall, and Feras Dayoub. Benchmarking sampling-based probabilistic object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[12] Tomoyuki Myojin, Shintaro Hashimoto, Kenji Mori, Keisuke Sugawara, and Naoki Ishihama. Improving reliability of object detection for lunar craters using monte carlo dropout. In *Lecture Notes in Computer Science*, pages 68–80. Springer International Publishing, 2019.

[13] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13991–14002, 2019.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[15] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[16] Pan Wei, John Ball, and Derek Anderson. Fusion of an ensemble of augmented image detectors for robust object detection. *Sensors*, 18(3):894, Mar. 2018.

[17] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[18] Haichao Zhang and Jianyu Wang. Towards adversarially robust object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 421–430, 2019.