Towards safer autonomous vehicles: Applying uncertainty quantification to pedestrian detectors

Bachelor's Project Thesis

Farrukh Baratov, s3927083 (f.baratov@student.rug.nl)
Supervisor: Dr. M.A Valdenegro-Toro

Abstract: Object detection networks have progressed at a rapid pace in recent years. However, detectors still suffer from many issues, such as a lack of network transparency and overconfidence in their predictions. Putting too much trust into unreliable predictions can have fatal consequences in safety-critical applications such as pedestrian detection for autonomous vehicles. A measure of network uncertainty can be implemented in order to give insight into the reliability of the detector. This study investigates how Monte Carlo Dropout, a method of uncertainty quantification, can be utilized to improve the performance of a Single-Shot Detector (SSD) network trained to detect pedestrians. We find that, based on the test dataset mAP scores of the detectors, the MC-Dropout model is able to improve its precision by about 7.47% compared to the baseline when it predictions with high uncertainty are suppressed.

1 Introduction

1.1 Background

Object detection has been the topic of much research in recent years. Developments such as YOLO [Redmon et al., 2015], Faster R-CNN [Ren et al., 2015], SSD [Liu et al., 2016] and RetinaNet [Lin et al., 2017] have been plentiful, rapidly advancing the discipline to new heights. The task of accurately detecting pedestrians in a real-world context is one application of object detection. Pedestrian detection is considered to be one of the "most crucial yet difficult issues" when it comes to developing intelligent vehicles, such as those capable of autonomous driving [Baek et al., 2014]. However, these networks still suffer from shortcomings. A study by Zhang et al. [2016] investigated the performance and limitations of the AlexNet and VGG object detectors on the Caltech Pedestrian Dataset. They found multiple sources of error, such as a failure to detect pedestrians that occupy a small number of pixels. This can have undesirable effects in safety-critical applications such as autonomous driving, where one false negative could have fatal consequences.

While most research aims to increase overall detector performance by designing precise models and effective training routines, it is also important to be able to accurately determine the "reliability" of individual predictions. Doing so would allow us to design trustworthy pedestrian detectors that can be safely deployed in real-world environments.

Uncertainty quantification (UQ) methods can be utilized in order to calculate predictive uncertainty, thereby allowing us to quantify prediction reliability. Recent studies have demonstrated the effectiveness of uncertainty quantification. One such study by Le et al. [2018] implemented two different aleatoric uncertainty estimation methods applied to a Single-Shot Detector (SSD). The results showed that both methods were able to successfully assign higher uncertainty values to false positives than to true positives. Furthermore, research by He et al. [2018] has found that their novel method of learning uncertainty quantification resulted in a 6% in Average Precision for the MS-COCO dataset.

Quantifying pedestrian detector uncertainty is essential for the purpose of "careful" models that are not overconfident in their own predictions. This would allow safer and more informed decisionmaking in potential real-world applications such as autonomous driving.

1.2 Research question

This leads to the main question of this study: How can uncertainty quantification techniques be used to improve the precision of pedestrian detection models? To answer this question, we use two SSDs: one "baseline" SSD that does not employ UQ, and one "dropout" SSD that utilizes MC-Dropout in order to measure predictive uncertainty in the form of the following values: bounding box mean and standard deviation, and classification score means. We believe that this will result in the following outcomes:

1. The dropout SSD is able to reliably assign higher standard deviation values to false detections compared to correct detections,

2. Filtering the dropout SSD detections based on bounding box uncertainty leads to a higher mean Average Precision (mAP) score compared to the baseline counterpart.

1.3 Contributions

The contributions of this study are as follows:

1. We provide an implementation of the SSD300 object detector that utilizes MC-Dropout, trained on a pedestrian dataset,

2. We analyse how implementing uncertainty quantification affects the performance of an SSD trained to detect pedestrians, provide insight into how utilizing uncertainty quantification for pedestrian detection can improve existing models.

2 Background

2.1 Caltech Pedestrian Dataset

The models are trained on the Caltech Pedestrian Dataset [Dollar et al., 2009]. The dataset contains around 10 hours of 30Hz video footage (approximately 250,000 individual frames) of urban traffic. Each frame is 640x480 pixels large. Each frame is annotated with bounding boxes for pedestrians, including pedestrians whose full bodies are occluded. There are four classes of pedestrians: one for single pedestrians, one for groups of pedestrians, one for possible pedestrians, and one for pedestrians that are far away.

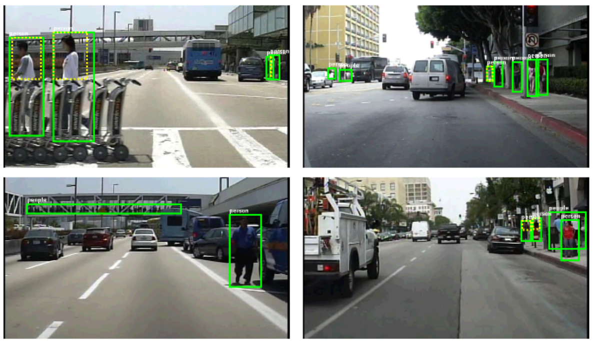

Figure 2.1: Annotated images from the Caltech Pedestrian Dataset. Green bounding boxes cover the pedestrian's whole body, while yellow annotations cover the part of the pedestrian that is visible. Image source: Dollar et al. [2009]

2.2 Single-Shot Detector

The Single Shot Detector (SSD) [Liu et al., 2016] is a one-stage object detector that predicts bounding boxes directly from feature maps, forgoing the need for a region proposal network. This allows the SSD to be more efficient and easier to train than two-stage counterparts. The network's output consists of 8732 bounding box proposals per class that are then filtered via non-maximum suppression (NMS). This results in up to 200 bounding boxes per class, each of which has a confidence score higher than a given threshold.

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \quad (2.1)$$

For its loss function, the SSD utilizes Equation 2.1, where N is the number of bounding boxes that have been matched to ground truths. This is a sum of the confidence loss $L_{conf}(x,c)$ and localization loss L_{loc}, which is weighted by the number of matched default boxes N. The confidence loss $L_{conf}(x,c)$ is a softmax loss over c categories. The localization loss $L_{loc}(c,l,g)$ is a Smooth $L1$ loss between the predicted box l and the ground truth box

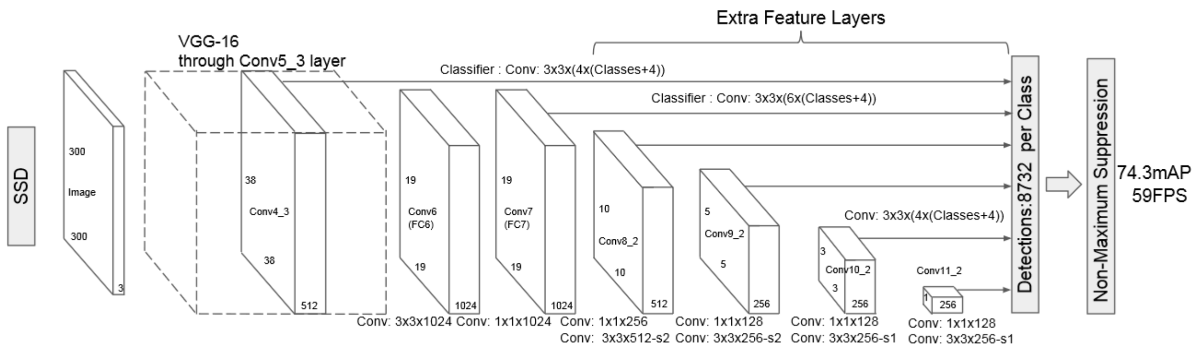

Figure 2.2: Architecture diagram of a Single-Shot Detector. The input is in the form of a 300x300 pixel image. The image is passed through multiple blocks in order to produce region proposals and classification scores. The output contains 8732 bounding box proposals and the associated classification scores per class. Image source: Liu et al. [2016]

g. The localization loss is multiplied by the balancing term α in order to ensure that the model does not focus too much on one task.

2.3 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) are a group of neural networks that adopt a Bayesian learning approach [Gawlikowski et al., 2021]. Given a set of inputs $X = \{x_0, x_1, ..., x_n\}$ and corresponding outputs $Y = \{y_0, y_1, ..., y_n\}$, we want to learn a predictive posterior distribution (PPD) of weights over data $p(\theta|X, Y)$ that likely generated the observations X and Y:

$$p(\theta|X, Y) = \frac{p(Y|X, \theta)p(\theta)}{p(Y|X)} \propto p(y|x, \theta)p(\theta) \quad (2.2)$$

Equation 2.2 displays the Bayes rule in the context of machine learning. This would also involve calculating the likelihood $p(Y|X, \theta)$ and evidence $p(Y|X)$. Given the PPD $p(\theta|X, Y)$, the output y^* for a novel input x^* can be estimated:

$$p(y^*|x^*, X, Y) = \int p(x^*|y^*, \theta)p(\theta|X, Y)d\theta \quad (2.3)$$

This method of prediction allows BNNs to encode uncertainty in their predictions. However, $p(y^*|x^*, X, Y)$ and $p(\theta|X, Y)$ are generally computationally intractable due to the intractability of the evidence term $P(Y|X)$, therefore we instead learn an approximate PPD $q(\theta|X, Y)$:

$$q(\theta|X, Y) \sim p(\theta|X, Y) \quad (2.4)$$

2.4 MC-Dropout

Monte Carlo Dropout (MC-Dropout) is a sampling-based method of uncertainty quantification and Bayesian approximation. It utilizes dropout [Srivastava et al., 2014], a method used to prevent overfitting a neural network model. This is done by putting a mask that randomly selects activations over a certain layer and sets them to zero. Traditionally, dropout is only utilized during training time. In MC-Dropout, the Dropout layers are also utilized during inference, resulting in stochastic predictions due to random activation selection. Gal and Ghahramani [2016] showed that sampling such a model allows us to calculate an approximate predictive distribution $q(y^*|x^*)$:

$$q(y^*|x^*) = \int p(x^*|y^*, \theta)q(\theta)d\theta \quad (2.5)$$

Sampling this distribution by performing N forward passes using the input x with the dropout-enabled model $f(x)$ allows the mean $\mu(x)$ and $\sigma^2(x)$ to be estimated:

$$\mu(x) = N^{-1} \sum_i f_i(x) \quad (2.6)$$

$$\sigma^2(x) = (N-1)^{-1} \sum_i (f_i(x) - \mu(x))^2 \quad (2.7)$$

where $f_i(x)$ is the model output at iteration i with model function f_i resulting from dropout. This results in an output distribution $f(x) \sim \mathcal{N}(\mu, \sigma^2)$, which encodes model predictive uncertainty as the variance.

3 System description

3.1 Data

We used the Caltech Pedestrian Dataset [Dollar et al., 2009] to train our models. To prepare the dataset for use with the PAZ SSD implementation, the dataset was converted from its initial format, utilizing .seq files for images and .vbb for annotations, into the PASCAL VOC format, which utilizes .jpg files for images and .txt files for annotations. The dataset was filtered in order to keep only images that contained ground truth bounding boxes. This resulted in a data-set of $\sim 122,150$ images.

The dataset is shuffled and split into three different sets: 70% of the dataset is used for training, 15% for testing, and 15% for validation. Dataset augmentation is performed on the training set and boxes in order to provide more “variety” to the training set, as many images in the dataset are part of one sequence, and therefore look very similar. The validation and test sets do not undergo augmentation.

3.2 Model implementation

The object detection model implementation used in this study are modified versions of the SSD300 implementation from the Perception for Autonomous Systems (PAZ) library [Arriaga et al., 2020]. This library proved to be a suitable choice for this study, as it allowed for easy installation and facilitated reliable training, validation, and testing of the models in question.

A single dropout SSD is trained in this study. This SSD is modified to contain dropout layers after each convolutional layer in blocks 6-9 of the network. These layers have a dropout rate of 0.3 and are enabled both during training and inference. As a result, the model produces stochastic predictions.

In order to create the baseline SSD, the trained dropout model's weights are copied to an SSD with an identical architecture. The new model's dropout rate is set to 0.0, resulting in deterministic predictions.

3.3 Detection pipeline

3.3.1 Baseline model

The baseline detection routine utilizes the PAZ `DetectSingleShot` function, which handles preprocessing, model predictions, and postprocessing. By default, it takes an RGB image as input for the model and outputs a set of bounding boxes for the image. The preprocessing consists of two steps: first, the image is converted from RGB to BGR. Afterwards, the image mean is subtracted from the image's BGR representation. Once this is done, the image is fed to the SSD as input.

$$pred_j = [x_j, y_j, w_j, h_j, c_{0_j}, c_{1_j}, ..., c_{n-1_j}] \tag{3.1}$$

$$out' = [pred_0, pred_1, ..., pred_{8731}] \tag{3.2}$$

For each input image, the model outputs a list of 8732 encoded (default) bounding box proposals in center format and their associated classification scores. The representations of the individual predictions and the list of predictions are shown in Equations 3.1 and 3.2 respectively. The position and dimensions of the box proposed by prediction j are given using their center form representation, which contains the center x-coordinate x_j, center y-coordinate y_j, the bounding box width w_j and height h_j. Each prediction has n classification scores $[c_{0_j}, c_{1_j}, ..., c_{n-1_j}]$ for classes $[0, 1, ..., n-1]$ associated to it.

The postprocessing begins with decoding the default box proposal coordinates and converting the boxes to corner format. In this format, the box coordinates are represented by $x_{min}, y_{min}, x_{max}, y_{max}$, the box's x-axis and y-axis extremes.

Afterwards, NMS is used in order to suppress boxes that overlap with an intersection-over-union (IoU) of more than 0.5. In these situations, the box with the higher confidence is preserved while the other is suppressed. Up to 200 boxes with the highest confidence are selected per class.

Finally, the selected boxes are filtered based on their confidence. For each class, all boxes that have a confidence lower than the specified threshold are removed from the selection, while the rest are saved as a part of the final set of predictions.

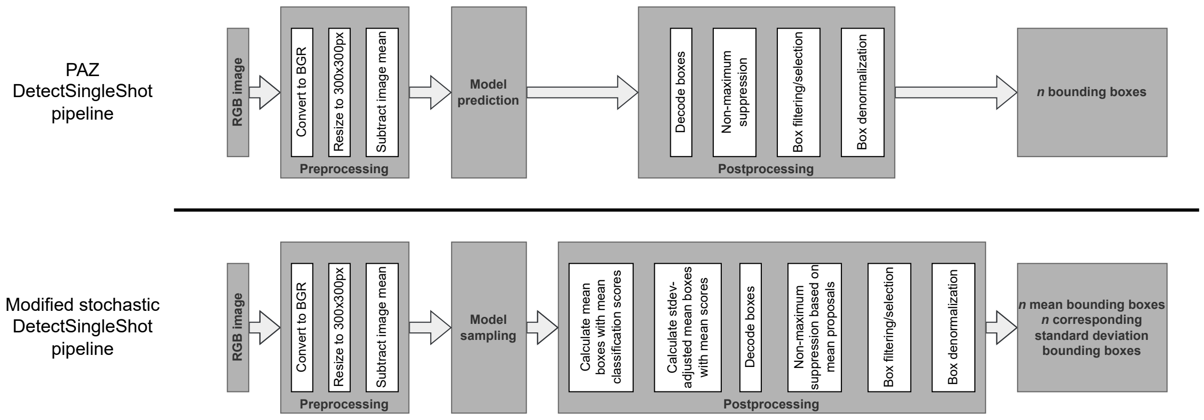

Figure 3.1: The PAZ SSD detection pipeline used by the baseline model (top) and the modified pipeline used for the dropout model (bottom). Both pipelines have the same input and preprocessing steps, however the modified pipeline instead samples the model and contains additional postprocessing steps for calculating mean boxes and scores, as well as standard deviation boxes. The NMS step is also modified such that NMS is performed on the mean and standard deviation boxes based on the mean boxes. The output contains two sets of bounding boxes, one for the mean and one for the standard deviation adjusted boxes.

3.3.2 Dropout model

The dropout model utilizes a modified version of the PAZ `DetectSingleShot` function. The preprocessing routine remains the same, but the input is fed into the network multiple times, necessitating a change in the postprocessing routine. The difference between the baseline and dropout pipelines are visualized in Figure 3.1, which depicts both prediction pipelines.

$$M_\mu = N^{-1} \sum_{i}^{N-1} M_i \tag{3.3}$$

The same input is fed into the stochastic network a total of N times. This results in N sets of 8732 proposals. In order to combine these sets, proposal coordinate means and standard deviations, as well as confidence score means, are calculated. Since the jth proposal in one set corresponds to the jth proposal in a different set, each set of proposals can be treated as a matrix M_i of dimensions $(8732, 4+n)$, with n being the number of classes including the background class. Equation 3.3 is used for calculating the means of each matrix, resulting in a set of predictions that contains the mean coordinates and classification scores of all N sets.

$$M_\sigma = \left(\frac{\sum_i^{N-1} (M_i - M_\mu)^2}{N} \right)^{\frac{1}{2}} \tag{3.4}$$

In order to calculate the standard deviation of proposals, a similar method of used where each set is represented as a matrix M_i. However, as we are only interested in the coordinate standard deviations, the classification scores are discarded from the matrix representation M_i. This results in M_i having the dimensions $(8732, 4)$. We use Equation 3.4 in order to calculate the set of coordinate standard deviations.

$$pred_{\mu+\sigma} = [x_\mu, y_\mu, w_{\mu+\sigma}, h_{\mu+\sigma}, c_{\mu_0}, c_{\mu_1}, ..., c_{\mu_{n-1}}] \tag{3.5}$$

$$w_{\mu+\sigma} = w_\mu + w_\sigma + x_\sigma \tag{3.6}$$

$$h_{\mu+\sigma} = h_\mu + h_\sigma + y_\sigma \tag{3.7}$$

σ-adjusted encoded predictions $pred_{\mu+\sigma}$ (Equation 3.5) are calculated in order to help visualize uncertainty. This is done by adding the standard deviations of the center x-coordinate and width to the mean width (Equation 3.6), and the standard deviations of the center y-coordinate and height to the mean height (Equation 3.7).

Both the mean and σ-adjusted predictions are then decoded independent of each other and the coordinates are converted to the corners representation, following the same procedure as the baseline model. Afterwards, NMS is performed on the boxes. NMS is performed based on the mean set, meaning that if the ith proposal of the mean set is suppressed, then so is the ith proposal of the σ-adjusted set. Afterwards, the boxes are filtered based on their confidence, as per the baseline model.

Additionally, a method of uncertainty-based suppression can be used to filter the boxes. All predictions where the IoU of the mean boxes and corresponding σ-adjusted boxes is below a certain threshold are removed from the dataset. This acts as a naive form of uncertainty-based false positive detection, where high uncertainty predictions are considered false positives and are therefore suppressed.

This concludes the postprocessing. The pipeline's final output comes in two equal-length sets of labelled bounding boxes with confidence scores: one set containing mean boxes and a set of σ-adjusted boxes corresponding to the mean boxes.

3.4 Training

A single model was trained the trained weights were used for the baseline and dropout models. The model's training routine utilized an SSD multibox loss function with default PAZ parameters and a balancing constant of $\alpha = 1$, and the stochastic gradient descent (SGD) optimizer with learning rate $\alpha = 0.001$ and momentum $\eta = 0.6$. The low momentum was selected because higher momentum caused loss to diverge.

The model was trained for a total of 30 epochs, using an NVIDIA RTX 3090 GPU.

The model's training and validation loss curves are visible in Figure 3.2. The curve shows that the model has converged or is close to converging. Another important observation is that the validation loss is lower than training loss for the majority of the training cycle, which is generally considered abnormal. This can be explained by the fact that dropout was used during training but not validation and testing, which would increase training loss, therefore making it higher than validation loss. Towards the final few epochs, validation loss increases above training loss.

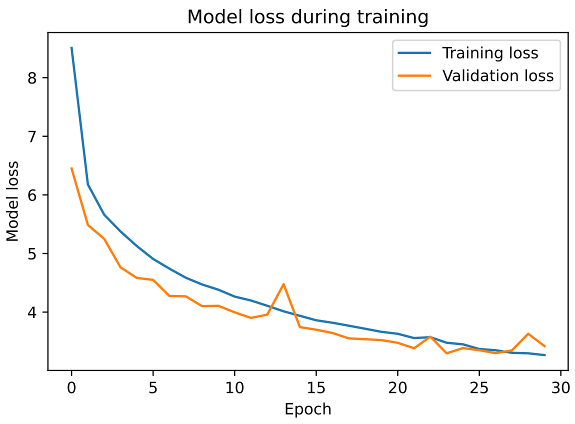

Figure 3.2: Training and validation loss for model. The dropout rate was set to 0.3 during training.

Figure 3.3 plots the curves of the individual components of training loss. It is visible that the negative classification component was consistently the lowest of all loss components, and positive classification was the largest. Furthermore, the total classification is significantly larger than localization loss during all epochs.

4 Results

4.1 Observations

Comparing the predictions (samples available in Figure 4.1 and Figures A.1, A.2, A.3, A.4 in Appendix A) made by the two models yields various observations. One observation is that both models make some predictions that appear to be complete nonsense - that is, they do not correlate to any observable features and seem to simply be results of model underfitting. Such predictions are especially prevalent when the input image contains a "busy" environment with many nearby pedestrians or cars.

The dropout model tends to assign very high uncertainty to false positives of this nature. However, false positives that still appear to detect objects, such as trees or mailboxes, do not always have high uncertainty. This indicates that high uncertainty is more indicative of false positives arising from model error than of false positives arising from detecting objects that are visually similar to pedestrians.

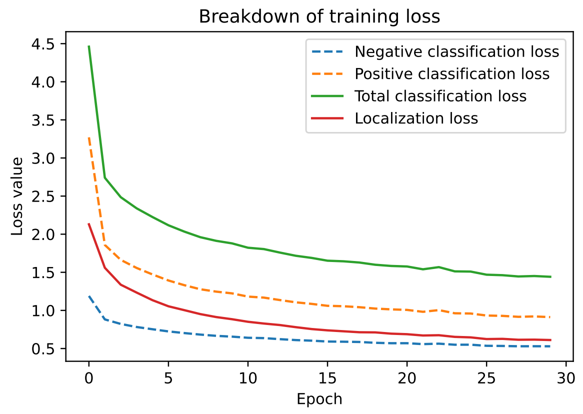

Figure 3.3: Values of training loss components per epoch. Total classification loss refers to the sum of positive and negative classification losses.

4.2 Precision

Model setup	person	person-fa	person?	people	mAP
Baseline	4.71%	5.11%	2.51%	13.0%	6.33%
Dropout-μ	5.50%	5.11%	2.51%	13.7%	6.71%
Dropout-filtered	**13.1%**	5.11%	2.51%	**34.6%**	**13.8%**

Table 4.1: The AP per class and mAP of each model setup. An IoU of above 0.5 is considered a match. "Dropout-μ" refers to the dropout model when assessed based on mean boxes without uncertainty-based suppression, and "Dropout-filtered" refers to the dropout model when high-uncertainty predictions are suppressed. Best per-column scores are in bold.

The precision of the models (Table 4.1) was assessed by calculating their AP per class and mAP on the test dataset. Three model setups were assessed: a deterministic baseline model, a stochastic dropout model that does not utilize uncertainty-based suppression, and a stochastic dropout model that utilizes uncertainty-based suppression. The detection processes used to evaluate mAP utilized an NMS IoU threshold of 0.45 and a confidence threshold of 0.5. Uncertainty-based suppression for the filtered dropout model utilizes an IoU threshold of 0.75. The mAP evaluation process considered boxes with an IoU of 0.5 or above to be matches.

Evaluation of model precision shows that the baseline has a low precision, with an mAP of only 6.33%. The baseline model has an AP of 4.71% for the class "person". It performed slightly better with faraway persons ("person-fa" label), and slightly worse with possible persons ("person?"). The model fares best when detecting groups of people, with 13.0% of its predictions being correct. This is expected, as groups of people would have larger bounding boxes than single persons, making it easier for the model to localize and classify them correctly.

The dropout model performs marginally better than the baseline, scoring 5.50% in the "person" class and 13.7% in the "people" class. Its AP for the other two classes is exactly the same as the baseline. This results in an mAP of 6.71%, only 0.38% above than the baseline.

The best precision comes from the dropout model with uncertainty-based suppression enabled. This model setup has an mAP of 13.8%, approximately 7.47% above the baseline. The model is most precise in the "person" and "people" classes, scoring 13.1% and 34.6% above the baseline respectively. The AP of the other two classes is the same as the baseline model.

The models having exactly the same AP for the "person" and "person-fa" classes is unsurprising, as both of these classes had very few samples in the dataset compared to the better-scoring classes.

5 Discussion

5.1 Hypothesis

The results of the mAP evaluation supports the hypothesis that UQ can be used to improve detector precision. While the performance difference between the baseline and unfiltered dropout models is trivial, filtering dropout results based on measured uncertainty causes a remarkable increase in model precision. Therefore, the results support the hypothesis outlined in Section 1.2: high uncertainty is indicative of false positives, and suppressing high-uncertainty predictions increases model performance in the context of pedestrian detection.

Alternatively, it is possible that the model's low precision is simply being compensated for by UQ, implying that higher-precision models would see diminishing returns from uncertainty-based suppression. This is supported by observations from test set predictions, where both the baseline and dropout models tend to detect multiple clearly false boxes

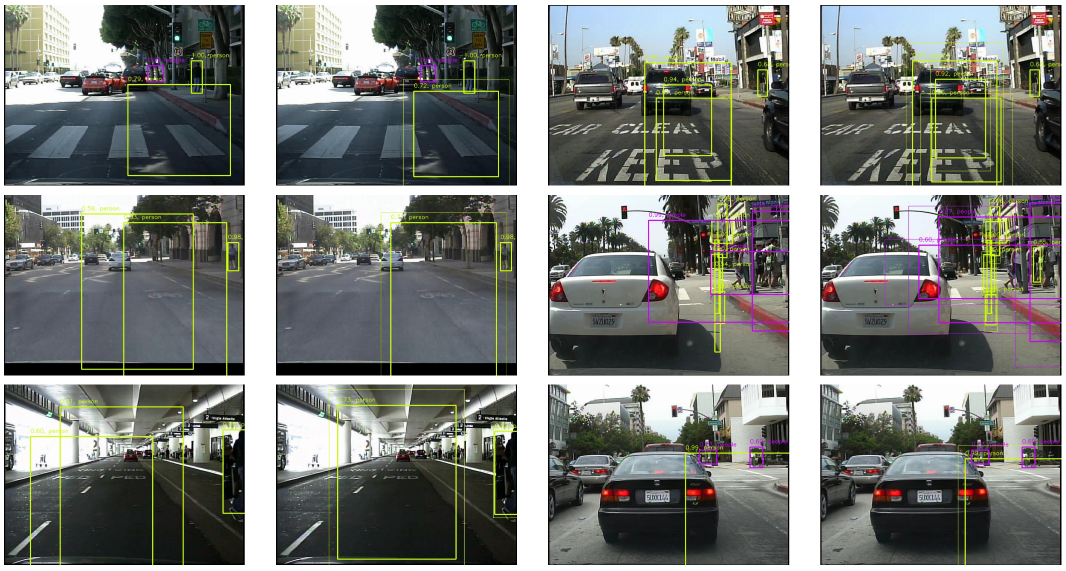

Figure 4.1: Pairs of predictions made by the baseline and dropout models on the same image. The dropout model's predictions contain two boxes: one thick-edged box to represent the mean, and one thin-edged box to visualize the model's localization uncertainty.

that do adequately localize any specific features. Currently, filtering predictions by removing high uncertainty boxes results in the eliminations of such boxes, causing a remarkable increase in precision. Assuming such predictions could be mostly eliminated via proper model and training routine design, we expect to see less drastic increases in precision due to uncertainty-based prediction filtering.

In order to accept or reject this explanation, the model setup and training routine would need to be revised and expanded in order to substantially improve the baseline detectors' performance and remove clear anomalies such as the "nonsense" false positives. The results would support the alternate explanation if they indicate that uncertainty-based filtering has a diminished effect on the precision when compared to the findings of this study.

5.2 Models

The models that we presented leave much room for improvement. Currently, most hyperparameters, are mainly selected based on the default values of the utilized implementations. Aside from that, certain hyperparameters were selected arbitrarily or based on available hardware. Furthermore, prior boxes utilized by the model were not tuned for this specific task at all. Therefore, tuning model hyperparameters would likely have a significant impact on the models and their performance.

The models are uncalibrated. This means that the confidence scores that the models provide with the boxes do not necessarily reflect the actual accuracy of the boxes. Since boxes are filtered based on confidence, this could lead to high accuracy boxes being suppressed or low accuracy boxes staying in the output.

Finally, the placement of dropout layers, and the selection of dropout probability, was selected mostly arbitrarily, with network structure and hardware taken being taken into consideration. Experimenting with dropout layer placement and probability could lead to improvements in performance for both models.

5.3 Training routine

While the training routine succeeded in delivering results, there is still much room for improvement. One issue is the presence of false positives that do not seem to be correlated to an actual object, which is likely because the loss function favors learning object classification over object localization. This is supported by Figure 3.3, which shows how localization loss is significantly lower than classification loss throughout training. This would explain why there are more nonsense bounding boxes in busy environments - the model detects object features, but fails to localize them properly. If this is the case, then the issue can be mitigated by adjusting the balancing constant α in order to give more priority to learning object localization.

In general, the hyperparameters for the loss function and optimizer were arbitrarily selected. Not only was α uncalibrated, but the learning rate and momentum of the SGD optimizer were intentionally set to lower values, as higher values caused the model to diverge to infinity. While the balancing constant can be tuned by trying different values to see what works best, the issue of diverging loss is less solvable, though it can once again be overcome by utilizing more powerful hardware that compensates for low learning rate by simply computing faster.

5.4 Dataset

The dataset was not ideally processed. One issue is that the dataset was created by shuffling and splitting the data rather than using the designated data splits that were outlined by Dollar et al.. This resulted in similar images appearing in the training and testing datasets, which could have impacted precision. However, training data was augmented before use, therefore this is unlikely.

Another issue is that the data used lacked any "negative" bounding boxes (bounding boxes labelled as background) or images with no non-background bounding boxes, but the loss function used indicated that they are present. This could have impacted model performance, as the training objective would be trying to minimize negative classification loss even though there are no negative examples present.

The aforementioned issues can easily be fixed by following the recommended data splits and including negative boxes in the training data and/or changing loss function parameters in order to accurately reflect the quantity of negative bounding boxes in the data.

6 Conclusion

In this paper, we proposed that UQ can be utilized to improve object detector in the context of pedestrian detection. In order to test this, we utilized two models, one deterministic SSD and one stochastic SSD that uses MC-Dropout to quantify uncertainty, both of which were trained on the Caltech Pedestrian Dataset. We found that, based on the test dataset mAP scores of the detectors, the MC-Dropout model improves its mAP by about 7.47% by suppressing high uncertainty prediction. The biggest observed improvement in AP was about 21.6% when detecting groups of people. Our findings indicate that UQ methods can be utilized in order to suppress false positives in pedestrian detectors based on measured uncertainty, thereby improving model precision.

We expect that this work will bring more attention to uncertainty quantification and its potential to detect false positives. We also expect that future studies will build upon our work by further investigating the potential applications of UQ methods in pedestrian detection and object detection in general, thus revealing new ways in which pedestrian detectors can made to be more precise, safer, and more reliable.

We hope our findings inspire others to investigate the extent to which UQ is effective in increasing pedestrian detection, as well as the situations in which uncertainty-based suppression is most effective. We additionally hope that future studies validate and generalize our findings by investigating the effectiveness of UQ methods in increasing model performance for different contexts, datasets and object detectors.

References

Octavio Arriaga, Matias Valdenegro-Toro, Mohandass Muthuraja, Sushma Devaramani, and Frank Kirchner. Perception for autonomous systems (paz), 2020.

Jeonghyun Baek, Sungjun Hong, Jisu Kim, and Euntai Kim. Bayesian learning of a search region for pedestrian detection. *Multimedia Tools and Applications*, 75:863–885, 2014.

Piotr Dollar, Christian Wojek, Bernt Schiele, and

Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2009. 10.1109/CVPR.2009.5206631.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/gal16.html`.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks. *CoRR*, abs/2107.03342, 2021. URL `https://arxiv.org/abs/2107.03342`.

Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection, 2018. URL `https://arxiv.org/abs/1809.08545`.

Michael Truong Le, Frederik Diehl, Thomas Brunner, and Alois Knol. Uncertainty estimation for deep neural object detectors in safety-critical applications. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3873–3878, 2018. 10.1109/ITSC.2018.8569637.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. 10.1109/ICCV.2017.324.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. URL `https://arxiv.org/abs/1506.02640`.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015. URL `https://arxiv.org/abs/1506.01497`.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. How far are we from solving pedestrian detection? *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1259–1967, 2016.

A Appendix

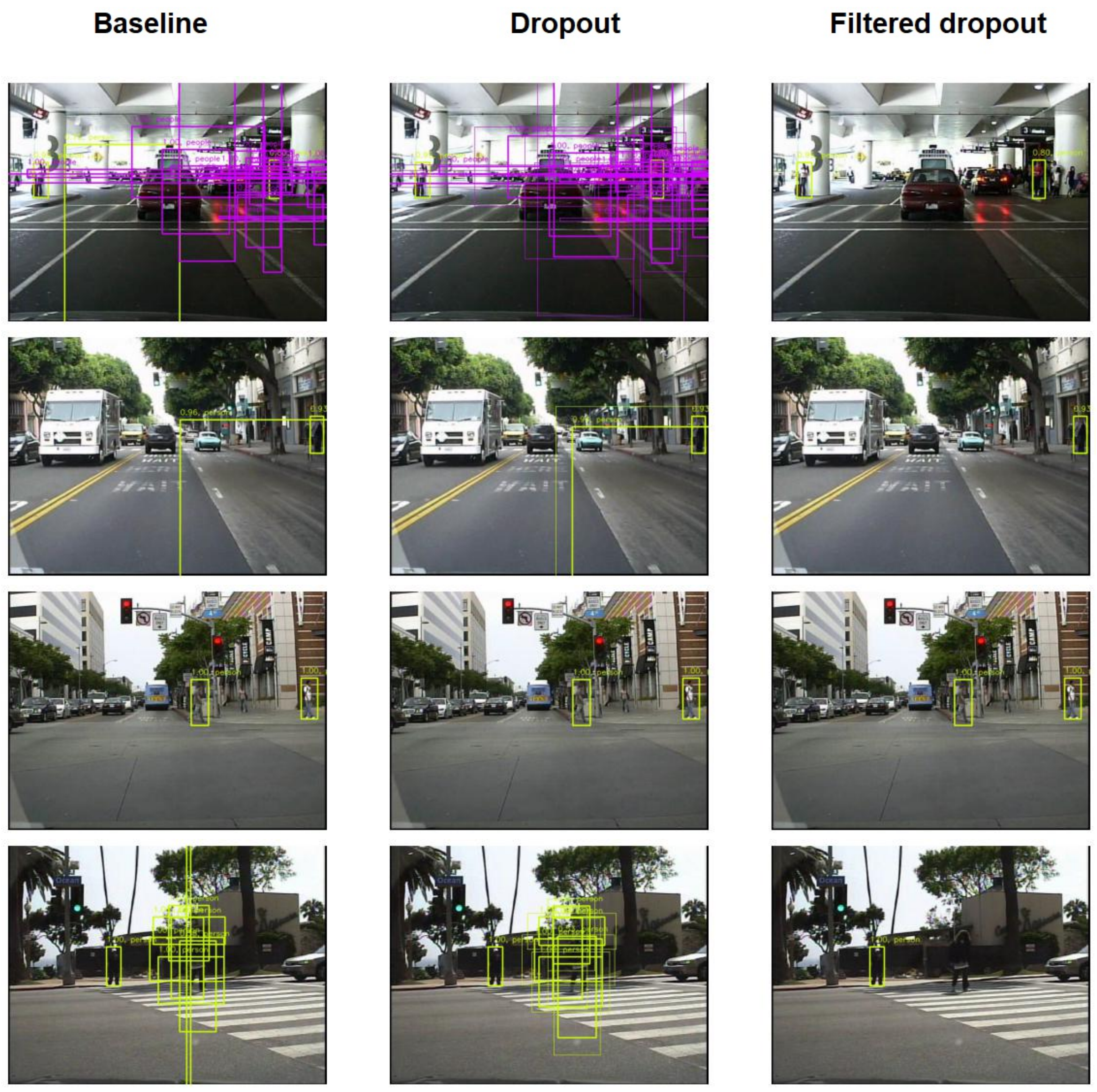

Figure A.1: Sample predictions on the same image per model setup. From left to right per row: baseline model predictions, dropout model predictions, dropout model predictions filtered based on uncertainty.

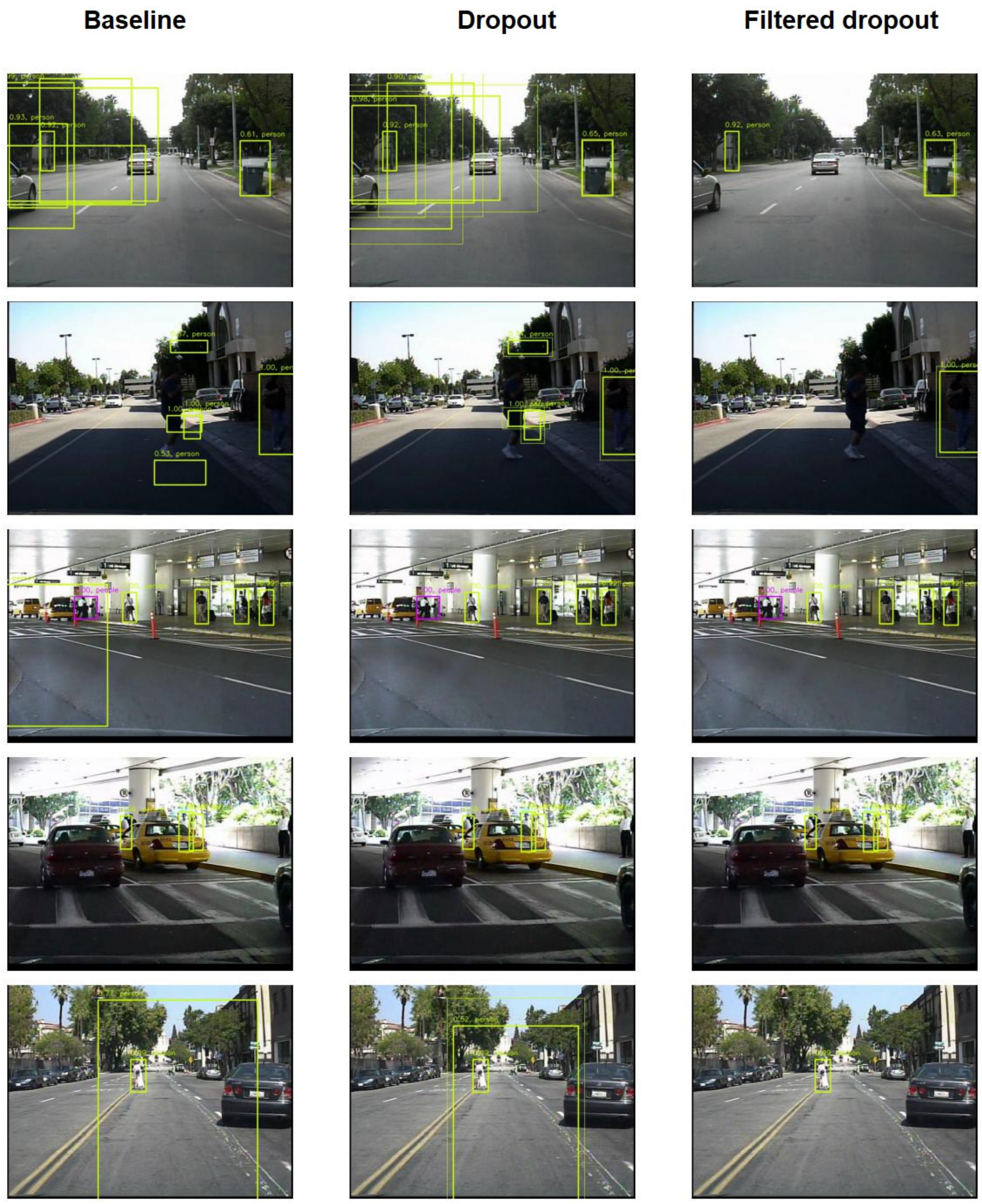

Figure A.2: Sample predictions on the same image per model setup. From left to right per row: baseline model predictions, dropout model predictions, dropout model predictions filtered based on uncertainty.

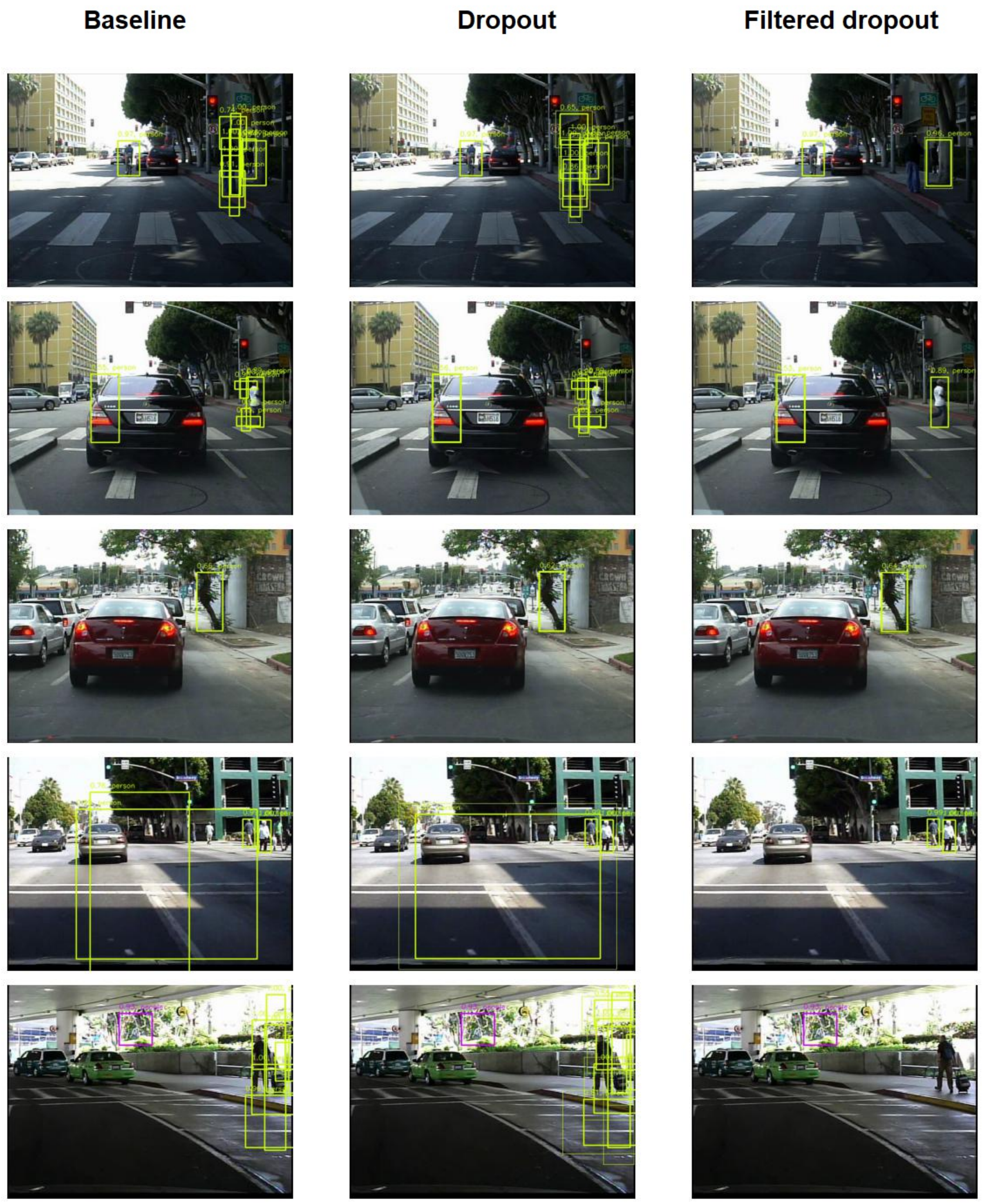

Figure A.3: Sample predictions on the same image per model setup. From left to right per row: baseline model predictions, dropout model predictions, dropout model predictions filtered based on uncertainty.

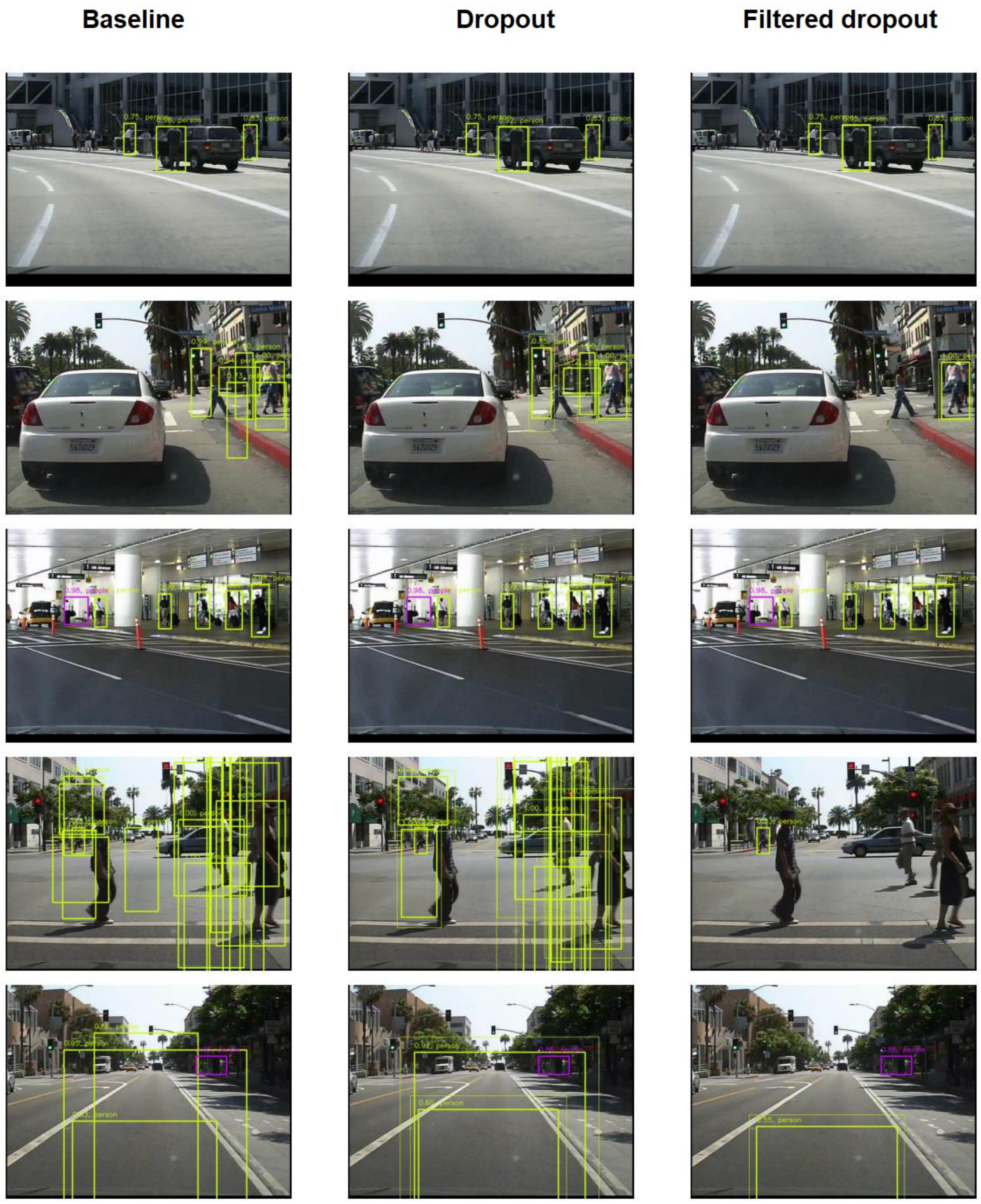

Figure A.4: Sample predictions on the same image per model setup. From left to right per row: baseline model predictions, dropout model predictions, dropout model predictions filtered based on uncertainty.