Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

German
Research Center
for Artificial
Intelligence

Master's Thesis

Benchmarking Out Of Distribution Detection Methods in 2D Object Detection

Jaswanth Bandlamudi

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G Plöger
Prof. Dr. Nico Hochgeschwender
Dr. Matias Valdenegro Toro
M.Sc. Octavio Arriaga

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Jaswanth Bandlamudi

Abstract

Autonomous Vehicles (AV) is one of the potential solutions to improve on-road safety. But AV are prone to making mistakes that might be catastrophic. These errors are the results of the perception models being confronted with unknown data or a novel event as these vehicles are deployed in open-ended environments. The unknown or novel data is generally referred to as Out Of Distribution (OOD) data. A method to detect this OOD samples is very important for the safe deployment of the AV.

This work reviews and investigates the ability and adaptability of various OOD detection methods in the task of object detection. Our approach is to regress a novelty score for every detection made by the object detector model. The novelty score is method-dependent, it can be a probability metric, a distance metric, or a descriptive statistical value.

We explored the abilities of Max Softmax, ODIN, and Uncertainty-based OOD detectors in detecting unknown objects and unknown environments. Since there are no established methods to evaluate the OOD detection in object detection. We proposed a new dataset called Out Of Distribution for Object Detection (OD^2) dataset for benchmarking. The dataset is made up of: BDD100K dataset which acts as an In Distribution (ID) dataset and is used for training our object detector, Indian Driving Dataset (IDD) dataset which acts as OOD dataset. We also used artificial climate images generated by ClimateGAN to test out-of-domain performance.

We found that the Bayesian model of SSD300 has outperformed the Sub-Ensemble model and vanilla SSD300 by 3.37% and 4.1% respectively in terms of object detection performance measured using mean average precision. But these methods struggled in performing OOD detection by producing almost similar AUROC scores compared to Max Softmax, ODIN, and uncertainty methods with probability, entropy, and box deviation as metrics. But Sub-Ensembles outperformed the Bayesian model by 11% in terms of AUROC scores while using the least number of parameters for uncertainty extraction purposes.

Acknowledgements

I would sincerely appreciate all of my supervisors for supporting me throughout my masters' studies and thesis period both technically and morally. The knowledge imparted by my professors at Hochschule Bonn Rhein Sieg has been boundless and second to none. Thanks for all the freedom and opportunities provided during this journey. I would also extend my appreciation to Iman, Alex, and Anastassia from Hochschule Bonn Rhein Sieg and Matias, Octavio from the German Research Center for Artificial Intelligence at Bremen for supporting me with the necessary infrastructure to accomplish my thesis work.

Huge credit is due to my Mother, Sister, Brother-in-law, and the rest of family for supporting me through my masters' studies. Their motivation and perseverance through all the personnel losses during this period have been humbling and made me the considerate human that I am today. No words are sufficient to thank my father and mother who by God's grace are my first teachers and role models for all the discipline, ethics, and values they imparted in my life.

The cities of Bonn and Cologne have provided me with a lot of memories. These cities are where I made many life-long friendships at the university and AKKA Technologies with Matthias, Santosh, Sasi, Lokesh, Ingrid, Ragith, Divin, and Alan. I really cannot thank you enough. I would like to extend my gratitude to all the front-line health workers, doctors, nurses, administration staff, and all the first responders during the corona pandemic times.

Contents

List of Figures

List of Tables

1

Introduction

Autonomous Vehicle (AV) perceives the operating environment using an array of sensors and navigate through any scenario without any human intervention. Though the development of AV is a highly sought-after challenge by many researchers and companies around the world. There are still many open questions surrounding the safety of the usage of AV. The safety issues stems from various stages in AV software pipeline as shown in Figure **??**.

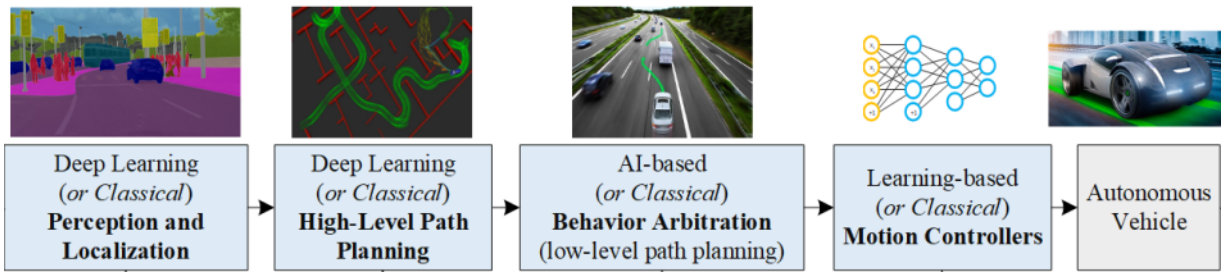


Figure 1.1: Autonomous driving stack adapted from the works of **?** , pg.4]. In the first stage, the operating environments are perceived and the agents are localized the information from this stage is used in the second stage to perform safe-path planning. The output from the path planning stage is the main input for lateral and longitudinal controllers to safely navigate through the complex world around them.

A dependable perception is very important for the safe operation of AV as the motion and path planning algorithms have heavily relied on the accurate localization of the objects in the operating environment. To extract the information needed for the safe operation of AV Deep Neural Networks (DNNs) are used. Since the operating environment is an open set [1], the ability to detect whether the sample is present in the training set helps in the safe implementations of AV.

DNNs which are trained discriminatively achieved best performances in tasks like speech recognition [**?**], object detection [**?**] and image classification [**?**]. Object detection is one of the tasks that is revolutionized with the application of Convolutional Neural Networks (CNNs) based DNNs [**?**]. It answers the following questions:

[1] The samples present during deployment can vary from the training dataset

1. what is in an image?
2. where is it?
3. how confident is the model in the detections?

Despite their imposing performance, DNNs based models are proved to be resulting in overconfident predictions when the model encounters data that is different from the limited domain of expected inputs due to noise, adversarial corruptions, or other changes in semantic distribution. In this work, we concentrate on the latter type of data often referred to as being Out Of Distribution (OOD) inputs [**?** **?**]. In the case of object detection, we consider the input is OOD if:

1. Input is outside the semantic space formed by the images used for training the perception algorithm.
2. Input consists of objects which are not used in training but have features closer to the object of interest.

To illustrate this failure as shown in Figure **??**, let us consider a model trained on Common Object in Context (COCO) object detection dataset [**?**]. We assume that the dataset is sampled from a distribution $p(X)$ at training time, the samples from the distribution $p(X)$ are considered to be in-distribution samples. Training an object detection model refers to modifying the model parameters thereby enabling the model to localize the objects in the image and classifying them into the object of interest. When the model is made to perform inference on the input sampled from Indian Driving Dataset (IDD) [**?**] which consists of novel objects like Auto-Rickshaw that are not present in the training dataset. As illustrated in Figure **??**, the network detects the novel object wrongly with a high probability of 81%. Figure **??** represents another OOD sample collected during a rainy day and the model couldn't detect the objects that are present in the image but detected a Dog that is not presented in the scene.

1.1 Motivation

The inherent behavior of DNNs in providing over-confident results when provided with OOD samples is unsolicited in safety-critical applications like medical diagnosis and autonomous driving. One such OOD input scenario is faced in Brazil by a Tesla vehicle operating in Autopilot mode [**?**], where a boy wearing an orange reflective (high-vis) jacket is detected and classified as an orange traffic cone. This erroneous behavior is dangerous since the true class of the input belongs to a movable object while it's classified to be an immovable object. This false detection further results in faulty predictions from motion prediction and planning algorithms which might result in serious injuries or fatalities.

Detecting and classifying this OOD samples results in the safe implementation of DNNs based object detection models where intentional or unintentional novel inputs may result in errors that might lead to catastrophic failures. Autonomous Vehicle (AV) development is one research area that benefits from the implementation of such OOD methods because of change in weather conditions, illumination levels, and also because of the novelty present in the semantics of the objects on the road. Detecting and localizing the OOD object present in the scene would help in better planning or fallback strategy.

(a) False Positive detection (b) False Negative detection

Figure 1.2: The sample image in Figure **??** represents False Positive detection in which an auto-rickshaw marked in blue is detected and classified as a car and Figure **??** represents False Negative detection in which cars are present in the scene but are not detected.

1.2 Challenges and Difficulties

- The majority of OOD detection methods proposed to date are designed entirely for object classification tasks. These methods proposed for object classification are not entirely adaptable to object detection tasks.
- The design philosophy of object detection problem includes foreground and background class. Background class is not generally included in training but should not be explicitly classified as a OOD data. The task of classifying between background class and OOD class is arduously challenging.
- Current available benchmarks are primarily proposed for evaluating OOD methods in classification and are comprised of small and simple datasets like CIFAR [**?**]. Approaches proposed using these benchmarks make them non-transferable to complex environments or datasets.

1.3 Problem Statement

In this work, we study the problem of out-of-distribution inputs in object detection tasks trained in a supervised setting. In particular, our task is to train a model using a dataset collected on the urban environment in clear weather and evaluate different OOD detection methods to detect objects in images sampled from OOD dataset.

The task formulation of our problem is to train a DNNs model M to perform regression task to extract the bounding box details together with a classification task to classify the type of object present in the image. We also employ another method or model to classify or quantify whether the input is OOD , we denote the model as M_{ood}. The deep learning model is trained using a dataset denoted by D_{train} and validated on a dataset denoted by D_{val} consisting of both In Distribution (ID) and OOD samples,

$D_{val} = D_{in-val} \bigcup D_{out-val}$. Finally, we test our model for benchmarking purposes on another dataset D_{test}, which is composed of both in-distribution and OOD samples $D_{test} = D_{in-test} \bigcup D_{out-test}$. Hence, an experiment E composes of M, M_{ood}, D_{train}, D_{in-val}, $D_{out-val}$, $D_{in-test}$, $D_{out-test}$.

The OOD detector should be able to produce a score which we refer to as ***Novelty Score (NS)***. NS can be a distance metric, a class-dependent probabilistic value, an entropy value or a descriptive statistic value based on the choice of the OOD method. Based on the method-specific score NS produced and a pre-defined threshold (δ) the OOD can be posed as a binary classification problem. A sample can be classified as ID or OOD as shown in Equation **??**.

$$X = \begin{cases} \text{In Distribution}, & \text{if } NS \geq \delta \\ \text{Out Of Distribution}, & \text{otherwise} \end{cases} \tag{1.1}$$

The behavior of the novelty score based on various inputs to the model is shown in Figure **??**

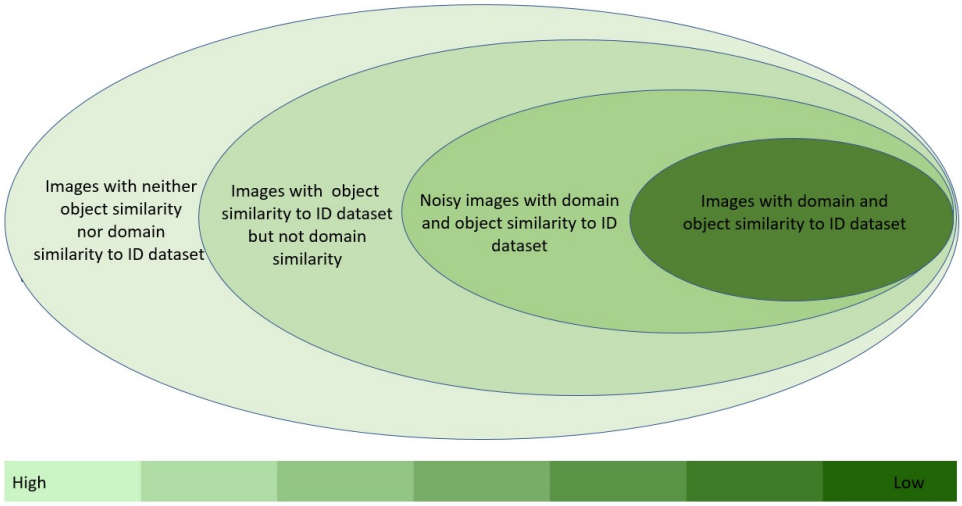


Figure 1.3: Expected behavior of novelty score based on the nature of the OOD input

For OOD dataset, as proposed by **?**] three distinct out-of-distribution categories are considered

- Dataset from a different domain. For example, data that is collected in an urban environment is considered as ID. While, the dataset collected in remote environment is used as OOD dataset.
- Dataset with poor quality features. For example, data collected in an environment with poor illumination, rain and snowing conditions.
- Dataset with inputs that are neither used nor prominent in the training data. For example, using KITTI dataset [**?**] as an in-distribution dataset and using IDD [**?**] which consists of novel unseen classes as OOD dataset.

2

Background

This chapter introduces basic knowledge that acts as a foundation for this thesis. We start by introducing the Neural Networks (NNs) which acts as basic computational units of deep learning. In this chapter, we introduce two different approaches that were used in this work to build and train NNs. In the first approach, we used Traditional NNs, in which the parameters of the neural network are modeled as single-point estimates. While in the second approach, we consider Bayesian approach in modelling the parameters of NNs which results in Bayesian Neural Networks (BNNs).

The methods in this thesis are either adapted or inspired from various works proposed by **?**], **?**], **?**], and **?**]. We provide a brief explanation of topics related to this work. For deeper knowledge and understanding we suggest the reader refer to the above-mentioned works.

2.1 SSD: Single Shot multi-box Detector model

Single Shot multi-box Detector proposed by **?**] is one of the well-performing models to solve the task of object detection. It is a single-stage object detection network that uses a feed-forward CNN to provide a default number of bounding boxes and their respective confidence scores for the objects present in the image. SSD is a fully-convolution network and consists of two components: a feature extractor and a multi-box regressor. In this work, the feature extractor is a truncated VGG19 model [**?**] by replacing all the fully-connected layers with convolutional layers followed by a multi-box regressor with several auxiliary convolutional layers. As the information from the final layers is coarse spatially, using these features affects the quality of object localization. Hence, the SSD performs object detection over multiple scales of multiple convolutional feature maps.

As the CNN reduces the spatial dimension gradually, the resolution of the feature maps also decreases. SSD uses lower resolution layers to detect larger-scale objects and vice-versa. Each convolution predictor in the SSD model produces a fixed set of predictions using the convolution filters . For a feature layer of shape $(m \times n)$ with c channels, the basic kernel for extracting the presence of a potential detection is a $(3 \times 3 \times c)$ shaped kernel. The features extracted can be used to extract either a category or a detection offset (ΔC_x, ΔC_y, Δw, Δh) at every feature map location. In SSD authors used default boxes, which are analogous to anchor boxes in Faster RCNN [**?**] and YOLO [**?**]. These default prior boxes encode a considerable amount of prior knowledge about the dataset in their respective shapes. The default boxes are designed to be placed at predefined locations and tiles the generated feature map in a convolutional

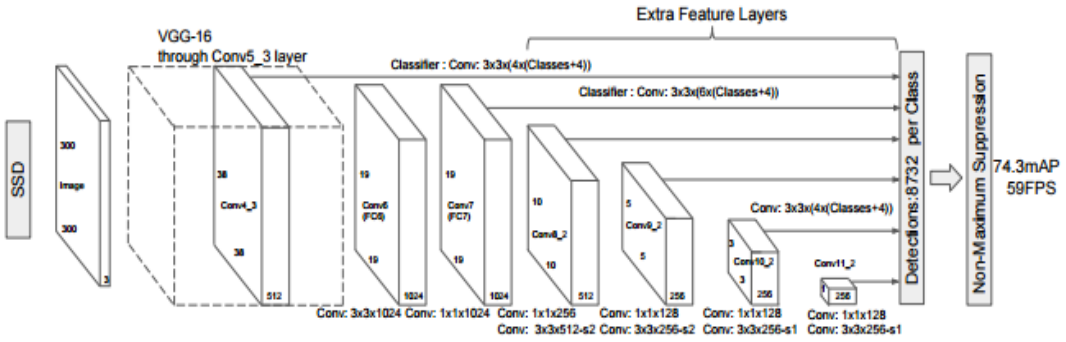


Figure 2.1: SSD framework proposed by **?** , p. 24]. The model with the stroked line is the feature extractor network based on VGG19 model and the feature layers are represented in solid lines.

format so that the position of each box relative to its corresponding cell is fixed. At every cell in the feature map, an offset relative to the default box shape and a class-specific score representing the presence of a class instance are predicted. At a given locations out of k boxes, a total of $(classes + 4) \times k$ filters are applied resulting in *(classes+4)*k*m*n* outputs for a $(m \times n)$ feature map. Different aspect ratios and scales for the default boxes result in the effective discretization of the possible bounding boxes.

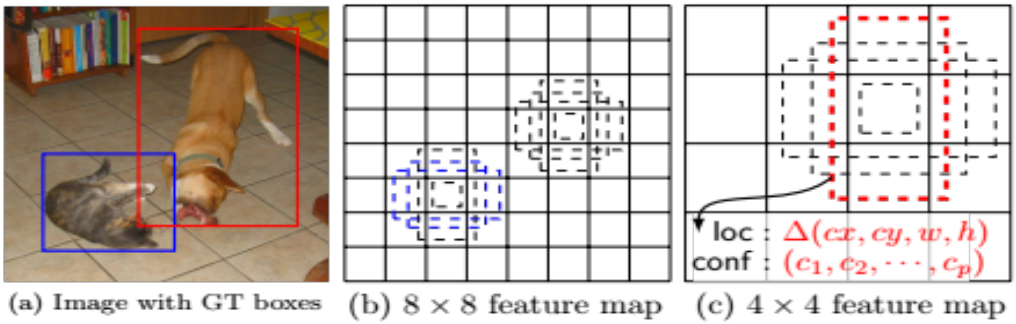


Figure 2.2: Prior boxes defined in different feature layers in SSD framework proposed by **?** , p.24]

Matching Strategy

Training an SSD object detection model needs an assignment of ground truth bounding boxes to a fixed size of regressed output boxes. Loss value is calculated using the assigned output boxes and ground truth information and a classical backpropagation are applied in an end-to-end fashion. SSD performance also is dependent on various other tasks like choosing a set of default box aspect ratios and the scales for detection, hard negative mining, and various data augmentation techniques.

We need to assign ground truth boxes to corresponding default boxes. The correspondence is found using Matching Strategy. The default boxes which vary in location, aspect ratio, and scale are matched to

ground truths with Jaccard overlap higher than a pre-set threshold. This poses the learning problem to determine multiple bounding boxes with high confidence instead of directly regressing the best fit box with maximum overlap.

Default box selection

The aspect ratios and scales of the prior box in a feature map are to be selected and assigned manually. This selection is dataset-dependent and also defined the number of default boxes in each image. The selection of prior boxes directly affects the performance of the SSD model since they directly affect the matching with the ground truth. The more the positive sample matching is high is the robustness and performance of the SSD object detection model.

Scales

The prior boxes at multiple scales targets objects of different sizes. In the vanilla SSD object detection model the minimum and maximum scales $(s_{\min}, s_{\max})$ are set to 0.2 and 0.9. The scale S_k of the k_{th} default box from m feature maps is computed using Equation **??**.

$$S_k = S_{\min} + \frac{S_{\max} - S_{\min}}{m - 1}(k - 1), k \in [1, m] \tag{2.1}$$

In the case of SSD300, the largest feature map is from $conv4_3$ layer and is assigned a scale of 0.2 and $conv9 - 2$ has the smallest feature map with a scale value of 0.9 and the rest of the layers has scales assigned evenly. Scale range can be tuned according to the scales of objects of interest present in the dataset.

Aspect ratios

Aspect ratio encodes a knowledge of different object shapes in the dataset. In the original work the aspect ratios are assigned as $a_r \in \left\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\right\}$ for all the multi-scale feature maps. Using the scale and aspect ratio values the width and height of the default box are calculated as $s_k\sqrt{a_r}, s_k/\sqrt{a_r}$ accordingly.

For an aspect ratio of 1 authors also added another scale $\sqrt{S_k S_{k+1}}$ to the bounding box. This resulted in a maximum of 6 default boxes at each cell. But, for optimizing the model and improving the inference speeds the first and final two feature maps are assigned with 4 bounding box ratios of 1,2, $\frac{1}{2}$. The aspect ratios and scales of the default boxes of the SSD300 model are shown in Table **??**.

Table 2.1: Table showing the number of default boxes for each multiple feature map layer. Also, observe that the first and last layers have reduced number of default boxes to increase inference speeds

no of box positions	38×38	19×19	10×10	5×5	3×3	1×1	total boxes
SSD300	4	6	6	6	4	4	8732

The optimal scales and aspect ratios are different for different datasets and should be re-designed for usage on a different dataset. As the dataset under consideration is BDD100k [**?**] and consists of a very

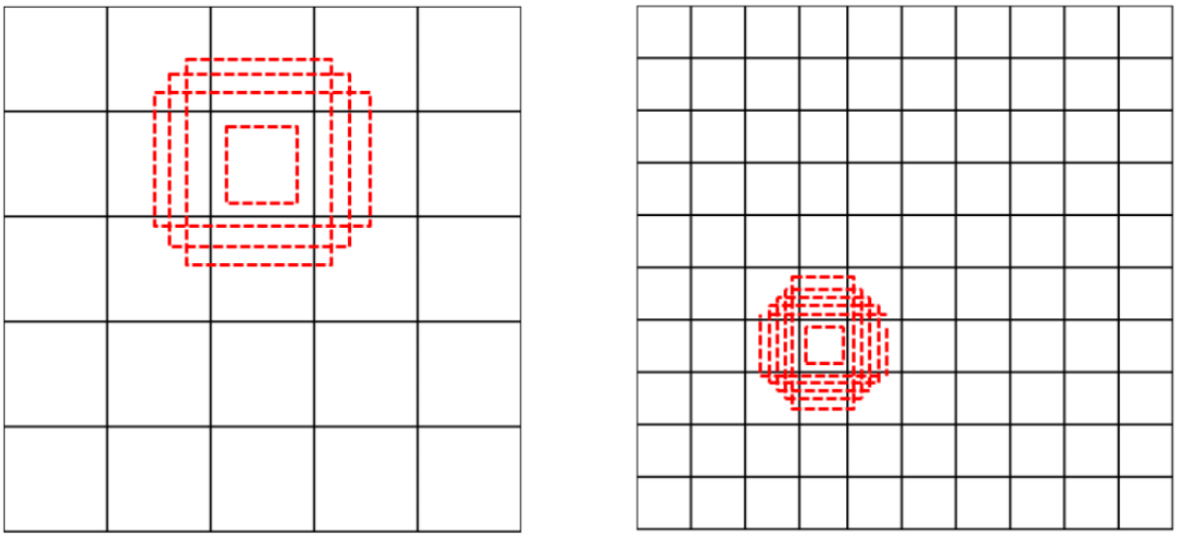


Figure 2.3: Example of default boxes placed for feature maps of size 5x5 and for feature maps of size 10x10

different class distribution in comparison to the VOC dataset. Hence the default boxes are to be tuned for better detection performance.

2.1.1 Loss Function

Deep learning-based object detection models solve classification and box regression tasks. Training a neural network involves reducing a loss value, which is the difference between the output of the model to the respective ground truth. In the case of SSD, this loss value is quantified using Equation **??**.

For a sample input image x, $x_{ij}^{p} = (1, 0)$ indicates a score for matching an i^{th} default box with j^{th} ground truth box belonging to a category p, l indicates a predicted box and g indicates a ground truth box

$$L(x, c, l, g) = \frac{1}{N}\left(L_{\text{conf}}\left(x, c\right) + \alpha L_{\text{loc}}\left(x, l, g\right)\right) \tag{2.2}$$

Hence, for object classification a confidence loss (L_{conf}) is considered.

$$L_{conf}(x, c) = -\sum_{i \in \text{ Pos}}^{N} x_{ij}^{p} \log\left(\hat{c}_{i}^{p}\right) - \sum_{i \in \text{ Neg}} \log\left(\hat{c}_{i}^{0}\right) \quad \text{where} \quad \hat{c}_{i}^{p} = \frac{\exp\left(c_{i}^{p}\right)}{\sum_{p} \exp\left(c_{i}^{p}\right)} \tag{2.3}$$

While for box regression a Smooth L1 distance between the regressed box and the respective ground truth boxes is used.

$$
\begin{aligned}
L_{loc}(x, l, g) &= \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \operatorname{smooth}_{\mathrm{L1}} \left(l_i^m - \hat{g}_j^m\right) \\
\hat{g}_j^{cx} &= \left(g_j^{cx} - d_i^{cx}\right) / d_i^w \\
\hat{g}_j^{cy} &= \left(g_j^{cy} - d_i^{cy}\right) / d_i^h \\
\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \quad \hat{g}_j^h &= \log\left(\frac{g_j^h}{d_i^h}\right) \\
\operatorname{smooth}_{L_1}(x) &= \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.4}
$$

2.2 Uncertainty Quantification in Deep Neural Networks

Uncertainty acts as a representation of the trustworthiness of a machine learning model, especially in the case of safety-critical applications. In the context of computer vision, **?**] proposed two major types of uncertainties, *Aleatoric uncertainty* that arise due to the inherent noise in the data like occlusions, motion blur in an image. *Epistemic uncertainty* is a result of a lack of knowledge on the data, it is observed when an object instance is under or not represented in the dataset used for training the models [**?** **?**]. *Aleatoric uncertainty* is an irreducible form of uncertainty, as the main reason for it is noise in collecting or querying the data [**?**]. But, the work proposed by **?**] proved that reduction in aleatoric uncertainty is made possible with noise-augmented data but not with lost information in the image. *Epistemic uncertainty* can be reduced by introducing data that might represent the testing conditions.

2.2.1 Bayesian Neural Networks

Deep Neural Networks since their inception are modelled with weights assumed to be point estimates [**?** **?** **?**]. This assumption makes the network more deterministic for provided inputs. To address these issues BNNs models the weights as a distribution over the provided dataset $P(w \mid D)$.As a consequence of probability distribution modelling w is considered as a random variable. This inherent randomness in NNs parameters enables the extraction of uncertainty from the NNs model.

Given a dataset $D = \left\{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right)\right\}$. The posterior distribution $p(w \mid D)$ can be obtained by applying Bayes theorem.

$$
p(w \mid D) = \frac{p(D \mid w) p(w)}{p(D)} \tag{2.5}
$$

The datasets available are very large resulting in the calculation of $p(D)$ being intractable [**?** **?**]. Hence, instead of exact modeling of $p(w \mid D)$ researchers proposed to approximate the eight distribution. The approximation can be done using various sampling algorithms like Metropolis-Hastings by **?**] and Markov-Chain-Monte-Marlo (MCMC) by **?**]. Though these methods can provide an unbiased

approximation, they are slow to converge. Researchers proposed another method called Variational Inference (VI) introduced by **?**] which though provides a slightly biased approximation but converges faster.

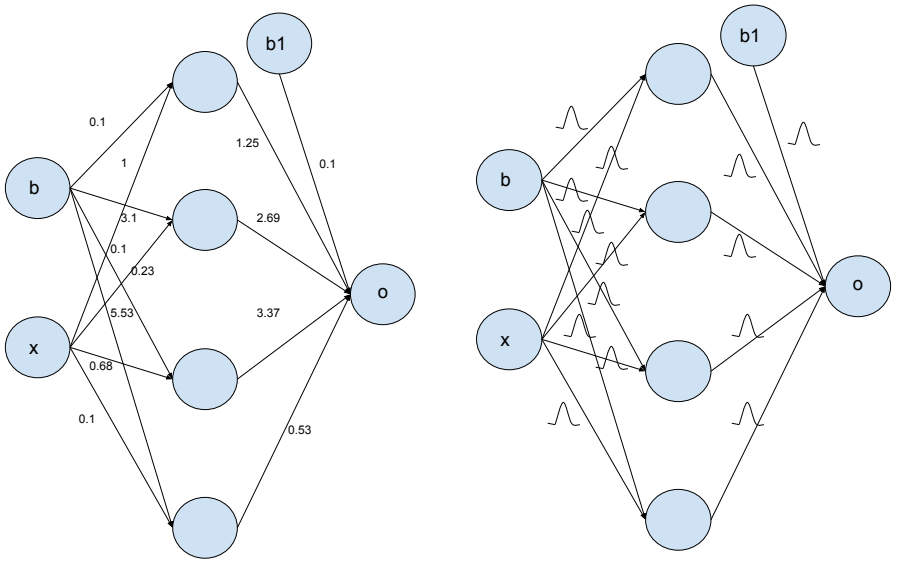


Figure 2.4: Representation of BNNs with the weights modelled as a gaussian distribution that is parameterized by mean (μ) and variance (σ) adapted from **?** , pg 3]

Using VI, $p(w \mid D)$ can be approximated to $q(w)$ parameterized with a mean (μ) and variance (σ). Using Bayes-by-Backprop proposed by **? ?**] optimize those distributions over the weight parameters to obtain the best approximation. The approximation is judged using Kullback-Leibler (KL) divergence introduced by **?**] as shown in Equation **??**.

$$\mathrm{KL}(q(x)\|p(x)) \equiv \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right] = \int q(x) \log \frac{q(x)}{p(x)} dx \tag{2.6}$$

The posterior calculation problem is now posed as an optimization problem of minimizing the KL-

divergence between q(w) and $p(w \mid D)$.

$$\begin{aligned}
\mathrm{KL}(q(w)\|p(w \mid D)) &= \mathbb{E}_{q(w)}\left[\log \frac{q(w)}{p(w \mid D)}\right] \\
&= \int q(w) \log \frac{q(w)}{p(w \mid D)} dw \\
&= \int q(w) \log \frac{q(w)p(D)}{p(D \mid w)p(w)} dw \\
&= \int q(w) \log \frac{q(w)}{p(w)} dw + \int q(w) \log p(D) dw - \int q(w) \log p(D \mid w) dw \\
&= \mathrm{KL}(q(w)\|p(w)) + \log p(D) - \mathbb{E}_{q(w)}[\log p(D \mid w)]
\end{aligned} \tag{2.7}$$

The optimal parameters θ^{opt} parameterized with a respective mean and variance (μ, σ) are defined as

$$\begin{aligned}
\theta^{opt} &= \mathrm{argmin}_\theta \, KL[q(\mathbf{w} \mid \theta)\|P(\mathbf{w} \mid \mathfrak{D})] \\
&= \mathrm{argmin}_\theta \int q(\mathbf{w} \mid \theta) \log \frac{q(\mathbf{w} \mid \theta)}{P(\mathbf{w}P(\mathfrak{D} \mid \mathbf{w}))} \\
&= \mathrm{argmin}_\theta \, \mathbf{KL}[q(\mathbf{w} \mid \theta)\|P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathfrak{D} \mid \mathbf{w})]
\end{aligned} \tag{2.8}$$

With the availability of $q(w)$, an approximation of $p(w \mid D)$ obtained. We can infer an output y^* for any given input x^* by integrating over the weights available as shown in Equation **??**.

$$p\left(y^* \mid x^*, D\right) \approx \int p\left(y^* \mid x^*, w\right) q(w) dw \approx \frac{1}{K} \sum_{k=1}^{K} p\left(y^* \mid x^*, w^k\right) \tag{2.9}$$

2.2.2 Deep Ensembles

Deep Ensembles proposed by **?**] are an ensemble of multiple deep learning models. All the ensembled models are with the same model architecture, each of these models is initialized randomly and trained using a dataset that is randomly shuffled. In general, an ensemble performs model combination in contrast to Bayesian Model Averaging. Hence, ensembles do not assume that a perfect model is present in the trained models. **?**] also investigated the usage of these proposed deep ensembles to estimate inherent uncertainty in the models.

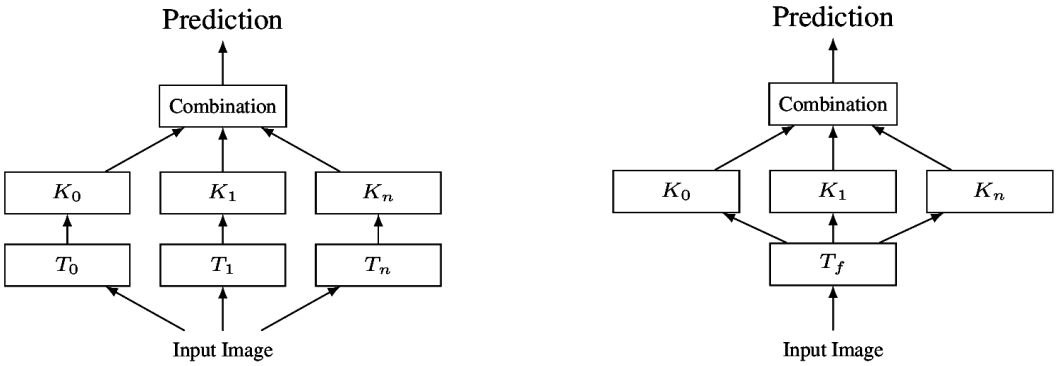
For an ensemble of size N and a model M, the schema used by **?**] is:

1. Each member of $\boldsymbol{N}$ is initialized with weights and biases independently
2. Train each member in the ensemble with randomly shuffled dataset
3. Final output from the ensemble is produced by the averaging model

Deep Sub-Ensembles

Deep Sub-Ensemble proposed by **?**] divides a neural network model into two sub-networks, the trunk network $\boldsymbol{T}$ and a task network $\boldsymbol{K}$. Hence, for an input x a neural network can be represented as $\boldsymbol{K(T(x))}$.

From the above representation, an Ensemble can be represented as $K_i(T_i(x))$, where i ∈ (1,...,N). From the work done by **?**], the author followed the work on Bootstrapped DQN [**?**] fixed the trunk network weights (T_f) and trained various instances of the task models $K_i(x)$, where i ∈ (1,..., N). In a nutshell, Deep Sub-Ensemble contains a fixed trunk network T_f and multiple instances of task network K_i which makes the ensembling process computationally less expensive and a real-time inference of both high-quality detection and uncertainty values is possible with GPU parallel computation ability.



(a) Ensemble

(b) Sub-Ensemble

Figure 2.5: Comparision of Ensembles and Sub-Ensembles, observe that there is only one trunk network in Sub-Ensembles which leads to a decrease in need for computational resources needed [**?** , p.2]

3

Related Work

This chapter provides an overview of the most common methods and architectures proposed to solve the task of object detection. It continues to furnish an outline about some Out Of Distribution (OOD) detection methods applied to solve classification tasks in computer vision.

3.1 Object Detection

In general, object detection is defined as *"determining whether are not the instances of objects present in the image and localizing them"*. The location of the objects is coarsely encoded as a bounding box *an axis-aligned rectangle tightly bounding the object of interest*. The task of solving object detection depends on each class of objects having features that are very distinctive from one class to another. These features are utilized further to find the interesting regions in the image consisting of the objects. Object detection methods are historically classified into two main categories: *traditional machine learning-based methods*, which uses carefully hand-crafted features and classify them using machine learning methods, and *deep learning-based methods*, which uses Convolutional Neural Networks (CNNs) to extract features by learning from experience. In this work, we majorly concentrated on deep learning-based object detection in a 2-Dimensional (2D) setting.

In deep learning-based 2D object detection, models are divided into region-based (Two Stage) and unified (Single Stage) models.

3.1.1 Region based (Two Stage) frameworks

The two-stage object detection model takes an RGB image as input. The model initially extracts category-independent region proposals using selective search or region proposal network. A classifier then process generated proposals to extract bounding box coordinates and objects class. The seminal works in two-stage object detectors belong to Regions with Convolutional Neural Networks (RCNN) family proposed [**?**].

RCNN proposed by **?**] integrates AlexNet [**?**] and selective search [**?**] to detect the objects in an image. Class independent region proposals that might contain any object of interest are proposed using selective search. Image is cropped and warped based into similar sizes based on the region proposals and are used to train a CNN model which is pre-trained for classification tasks on another dataset like

ImageNet [**?**]. Class-specific Support Vector Machines (SVMs) are trained using the fixed-length features extracted using CNN. This SVM replaces the softmax layer present in the CNN. Class-specific bounding box regressor extracts bounding box values from the features extracted by the CNN. Though RCNN was able to outperform the existing models in terms of detection quality it suffered from many drawbacks. More notably, training an RCNN model is slow and hard to optimize due to the need for training various sub-models independently. Also, the amount of computational and memory requirements are high due to the requirement for features needed to train the SVM are to be stored. Also, the inference time is high due to the extraction of per object proposal for every test image.

To address the disadvantages in the RCNN model, another method Fast RCNN is proposed by **?**]. In this model whole image is processed by CNN to generate object proposals. A feature vector is extracted from each of the proposals generated using a Region Of Interest (ROI) pooling layer this essentially acts as a learned warping function at the feature level. The extracted features are passed through a stack of Fully Connected (FC) layers and divided into two different heads, one with a softmax layer which acts as a classifier and another head extracts class-specific bounding box offsets which are used for the further refinement of object proposals. The most important step that the Fast RCNN model introduced is the ability to train the object detection network in an end-to-end manner. This model resulted in reducing the training time by three-fold while the inference speeds are improved by 10 times. Another most important optimization is done in memory required with the removal of feature caching required for training the RCNN model. Though Faster RCNN sped up the training and inference speeds by multiple folds, it still depended on the externally generated region proposals which are exposed as a bottleneck for real-time applications.

Faster RCNN is a third-generation RCNN model proposed by **?**] which uses the CNNs ability to localize objects using convolutional layers. Hence, Faster RCNN used an efficient Region Proposal Network (RPN) for generating region proposals. RPN is shared for region proposal generation and also for performing a classification task. RPN initially produces k boxes referred to as *anchor boxes*, which are of different scales and aspect ratios at each Convolutional feature map location. These anchor boxes are generated independent of the image content but the features generated by the convolution layers from these anchor boxes are image content dependent. Each feature extracted from an anchor box is fed to a stack of FC layers which is further branched into a classification network and a box regression network.

3.1.2 Unified (Single Stage) frameworks

The region-based frameworks are computationally expensive in devices with less storage or computational ability. This drawback in region-based networks had accelerated the research in unified frameworks. The unified framework predicts the object class and regresses the bounding box offsets from the input image in a single forward-pass. These models avoid the usage of region proposal generation or feature post-processing.

The pioneering work in unified frameworks is You Only Look Once (YOLO) proposed by **?**], which divides an image into a 7 x 7 grid and performs classification and localization on each cell in parallel. It is observed that framing the object detection problem as a regression problem resulted in reduced processing

time. It outperformed the Deformable Part Model (DPM) proposed by **?**] and R-CNN **?**]. This method struggle from not being able to detect objects of smaller sizes. There were revisions done to the initial YOLO models proposed by the same authors and proposed YOLOv2 and YOLOv3, though there are improvements in detection accuracy it is not considerable. A considerable update is done in YOLOv4 proposed **?**] in which authors used different strategies in data augmentation (Mosaic, CutMix, Label smoothing), loss functions for bounding box (IoU loss, GIoU loss, DIoU loss, CIoU loss), and activation functions (leaky-ReLU, Swish, Mish). The model trained using the above advancements led to an increase in the Average-Precision (AP) values from 38.0% to 42.4%.

Single Shot Multi-box Detector **?**] is another single-stage object detector model in which object detection is done at several different layers of CNN instead of only doing it from the last layer. This technique allows us to detect objects of different sizes and also improved the accuracy predominantly. The current State-Of-The-Art (SOTA) object detector is EfficientDet proposed by **?**]. The authors proposed Bi-directional Feature Pyramid Network (Bi-FPN) in which a modified feature pyramid network proposed by **?**] is used as a base network and is modified by varying the connections in Bi-FPN to modify the intermediate features to consider thereby increasing the accuracy along with several parameters that the model constitute. The authors also proposed a compound scaling method, which performs the up-scaling of resolution for regression of bounding box and objects classification. Eight different EfficientDet networks are starting from D0 to D7, which is controlled with compound scaling coefficient ϕ.

3.1.3 Object-Detection Metrics

In this section, we present various metrics that are used to measure the performance of the object detectors.

Intersection over Union

Intersection over Union (IoU) is an evaluation metric that is employed to measure the detection accuracy of an object detector. To apply this metric we need ground-truth encoding of the bounding box and the encoding label of the bounding box. IoU is computed using Equation **??**, as the area of intersection between the predicted bounding box and ground-truth bounding box divided by the area from the union of the boxes. An IoU value of unity implies the two bounding boxes are identical.

$$IoU = \frac{Area\ of\ intersection\ of\ predicted\ and\ ground\ truth\ boxes}{Area\ of\ union\ of\ predicted\ and\ ground\ truth\ boxes} \tag{3.1}$$

Precision and Recall

Precision refers to the percentage of your results that are relevant and Recall refers to the percentage of total relevant results correctly classified by the model. An object detector is performing well if its precision is high with an increase in recall value. Precision and Recall are calculated using True Positive, False Positive, and False Negative rates.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Average Precision

Average precision summarizes the precision value for 11 equally spaced recall values i,e. [0, 0.1, 0.2,.....,1]. The precision for each recall level is calculated as the maximum precision value measured where the corresponding recall is greater than or equal to the corresponding recall value.

$$\text{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{\text{interp}}\left(r\right) \tag{3.4}$$

$$p_{\text{interp}}\left(r\right) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{3.5}$$

Mean Average Precision

The metric mean Average Precision (mAP) is introduced to evaluate the performance of object detectors in which there is more than one foreground class. mAP is the mean of the per-class AP scores.

3.2 Out-Of-Distribution Detection

As mentioned in Section **??** the majority of well-performing object detection methods are learning-based. But, the majority of the existing learning-based models are evaluated on the assumption that the test data are drawn from independent and identical distribution (*i.i.d*) also known as In-Distribution (ID). However, in a real-world deployment of these models might need to infer samples that can be Out-Of-Distribution (OOD). The distribution shift between OOD and ID stems from either *semantic shift* (samples from different class) or *covariate shift* (samples from a different domain).

Class setting in OOD detection is visualized in Figure **??** and consists of following problems

- **Sensory anomaly detection**: the samples with covariate shift from train images are considered OOD. No semantic shift can be observed.
- **Semantic anomaly detection or one-class novelty detection**: the sample images with semantic shift will be considered as OOD. No covariate shift is observed.
- **Multi-Class novel detection**: ID images consists of multiple classes to detect. while, samples with semantic shift will be considered OOD. No covariate shift is observed in the test samples.
- **Open-set recognition**: is a similar setting as multi-class novelty detection but includes classification of classes in in-distribution samples.

- **Out of Distribution detection**: OOD detection is a super-set which includes semantic anomaly detection, one-class novelty detection, multi-class novelty detection and open-set recognition that aims at detecting samples with semantic-shift without deteriorating its performance on ID dataset.

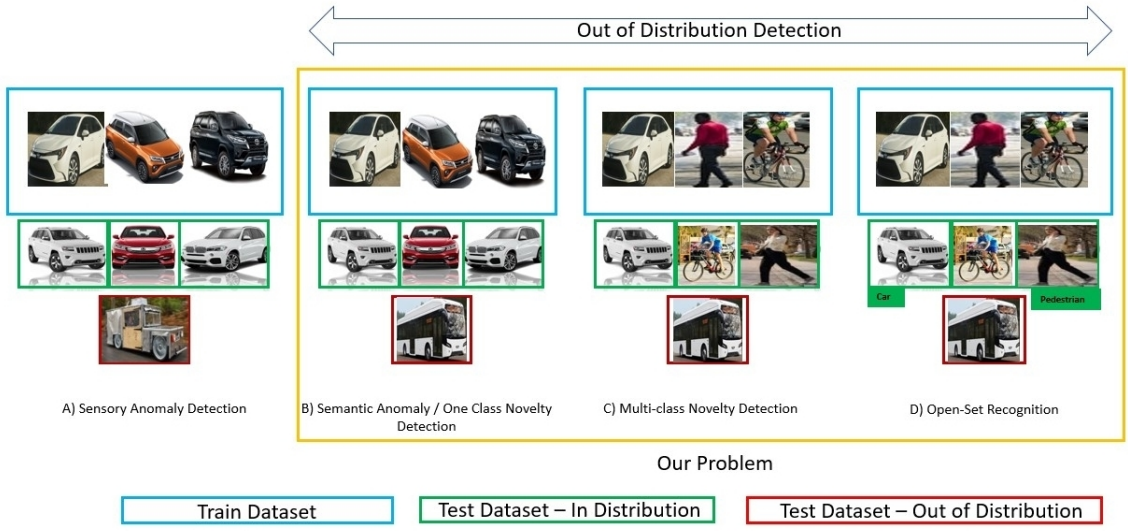


Figure 3.1: Class differentiation in generalized OOD detection framework

Out-of-distribution detection methods classify the inputs that are sampled from another distribution from the inputs used for training the inference model. These methods can be classified into four different groups: Metric, Inconsistency, Generative, and Uncertainty based approaches.

3.2.1 Metric based methods

Metric methods determine if the input is OOD based on the behavior of the current input following the behavior of the samples inside the generalization area of the DNN under investigation.

One of the works done is by **?**] in which a classification model is modified to output an additional value c along with a softmax score. The training pipeline is modified to include a two-folded loss while penalizing a misclassification using cross-entropy loss along with a penalty for lower confidence score c. Another method proposed by **?**] in which intermediate values like layerwise norm, minimum and maximum value of the layer-wise gradient of the weights with respect to the loss function of the predicted class are fed to a logistic regressor along with entropy of the output class. The logistic regressor produces a score value which helps in deciding whether an input is OOD.

?] also adapts the training procedure of the classification model. The authors extract a carefully curated dataset that contains samples outside the generalization area and assign a uniform distribution of values over all the classes as the labels and are used while training along with the original training dataset. An input leading to output with higher entropy is judged to be an OOD sample. Another work

proposed by **?**] computes Mahalanobis distance between activation space of the DNN layers to the closest class-conditional Gaussian distribution. All the layerwise distances are fed to a logistic regression network which outputs a score value to decide whether the input is an OOD input.

3.2.2 Inconsistency based methods

The basic idea behind inconsistency-based methods is to induce a minimal change to the sample from the in-distribution dataset. During inference, both the image and the modified image are processed and the distance between the outputs is used to classify the in-distribution sample and OOD sample.

ODIN proposed by **?**] in which the OOD detection is done by scaling the logit layers before the softmax output by a constant value called temperature, as well as perturbs the image input by a small amount. A sample is expected to be in distribution if the output value of the modified image for the original predicted class is high. These methods had resulted in the best performing model for detecting OOD data points.

3.2.3 Generative methods

Generative methods are similar to inconsistency-based methods except in the way perturbation to the input images is performed. The perturbation is made to shift the image more towards the training distribution and is made using a generative network trained on the in-distribution training dataset. During inference, both the original and the generated image are processed, and the higher the distance between both outputs the confident we are that the input sample is OOD.

One method proposed using this philosophy is proposed by **?**] in which an encoder is employed on top of the penultimate layer of the original DNN, which helps in reconstructing the image. The difference between the original image and the generated image along with the output of the final layer and the penultimate layer to an abnormality module which regresses the confidence score that can be used to differentiate OOD input from an in-distribution input. Another such work is proposed by **?**] in which the authors used likelihood ratio value. if the likelihood ratio determined between the output of the model trained on the original images and the one on the perturbed images is low, the input image is expected to be OOD. The authors had trained two generative PixelCNN models by **?**], one on the original dataset and another on the slightly perturbed images from the training dataset. From the observation, they concluded that the model trained on original images is sensitive to the true class contents compared to the model trained on perturbed data. Hence, a low likelihood ratio value points to an OOD input.

3.2.4 Uncertainty based methods

An uncertainty measure could be directly applied to reject OOD samples as we would expect the uncertainty to be high on such inputs. There are multiple uncertainty types coined in various researches proposed by **? ? ?**]. The uncertainty extracted using Bayesian approaches is called epistemic uncertainty which measures the uncertainty in estimating the model parameters given the training data. In other

words, epistemic uncertainty measures how well the model is matched to the data in terms of model structure and parameters. The other type of uncertainty is aleatoric uncertainty, which is irreducible uncertainty that arises from the natural complexity of the data, e.g. from class overlap or label noise. Some works presented by **?** **?** **?**] had presented an argument that OOD data is implicitly modeled by epistemic uncertainty.

Exploiting this idea, **?**] proposed the usage of an ensemble of multiple models to extract epistemic uncertainty which can be used to judge whether an input is from in-distribution. Similarly, **?**] proposed prior networks for uncertainty estimation and dis-entangle the predictive uncertainty into three contributing factors model uncertainty, data uncertainty, and distributional uncertainty. The authors used distributional uncertainty, which arises when input is sampled from a distribution that is very different from the training distribution.

3.3 Out Of Distribution methods - Deep Dive

In this section, we provide background information about various classical OOD detection methods used in this thesis. This section differs from Section **??** in providing deep-dive information in explaining why the OOD detector works.

3.3.1 Max Softmax Probability

The work proposed by **?**] proposed a metric to separate the In-Distribution (ID) sample from the Out-Of-Distribution (OOD) sample which uses maximum softmax confidence score from the softmax layer of the network. The authors' reasoning in using the softmax score as a metric is though the network yields over-confident scores for the OOD sample. But these confidence values are often lower than the probability with which ID samples are classified.

Following the above reasoning, the authors used the Equation **??** to obtain a maximum value of softmax scores which can be used as a novelty score to classify an ID and OOD sample.

$$s\left(\mathbf{x}^{*}\right)=\max_{c} P\left(y_{c} \mid \mathbf{x}^{*}; \mathcal{D}\right) \tag{3.6}$$

In Equation **??**, $\mathbf{x}^{*}$ is the sample that belongs either to an ID class or to an OOD class, y_c is the class label, and $P\left(y_{c} \mid \mathbf{x}^{*}; \mathcal{D}\right)$ is an estimation of class specific probabilities from a Neural Network (NN) trained on dataset $\mathcal{D}$

3.3.2 ODIN

ODIN is a post-hoc method proposed by **?**], it is mentioned as a post-hoc method as there is no modification of training is needed to separate ID and OOD samples. This method out-performed Max Softmax Probability by using the scaling of logit layers before the softmax layer of a neural network by a Temperature value **T** along with a small gradient-based perturbation added to the input image.

In this work, the author used images from CIFAR-10 and CIFAR-100 as ID samples and images from TinyImageNet, LSUN, and artificially generated datasets made up of Uniform and Gaussian noise.

Temperature Scaling

In previous works, Temperature scaling is a method used in distilling the knowledge in neural networks by **?**] and for calibrating the confidence scores in classification tasks by [**?**]. ODIN proposed by **?**] observed from various experimentation that temperature scaling also can be used to separate ID and OOD samples.

For a neural network **f** that is trained on a dataset for multi-class classification. For an input **x**, we calculate softmax scores using the Equation **??**. The class with maximum softmax probability $S_{\hat{y}}(\boldsymbol{x};T) = \max_i S_i(\boldsymbol{x};T)$ is used as class output.

$$S_i(\boldsymbol{x};T) = \frac{\exp\left(f_i(\boldsymbol{x})/T\right)}{\sum_{j=1}^{N}\exp\left(f_j(\boldsymbol{x})/T\right)} \tag{3.7}$$

To show the effectiveness of Temperature we perform Taylor expansion of the softmax function presented in Equation **??**,

$$\begin{aligned} S_{\hat{y}}(\boldsymbol{x};T) &= \frac{\exp\left(f_{\hat{y}}(\boldsymbol{x})/T\right)}{\sum_{i=1}^{N}\exp\left(f_i(\boldsymbol{x})/T\right)} \\ &= \frac{1}{\sum_{i=1}^{N}\exp\left(\frac{f_i(\boldsymbol{x})-f_y(\boldsymbol{x})}{T}\right)} \\ &\text{performing Taylor expansion and neglecting the terms from } 3^{rd} \text{ order and above, we get} \\ &= \frac{1}{\sum_{i=1}^{N}\left[1+\frac{f_i(\boldsymbol{x})-f_j(\boldsymbol{x})}{T}+\frac{1}{2!}\frac{\left(f_i(\boldsymbol{x})-f_j(\boldsymbol{x})\right)^2}{T^2}+o\left(\frac{1}{T^2}\right)\right]} \\ &\approx \frac{1}{N-\frac{1}{T}\sum_{i=1}^{N}\left[f_{\hat{y}}(\boldsymbol{x})-f_i(\boldsymbol{x})\right]+\frac{1}{2T^2}\sum_{i=1}^{N}\left[f_i(\boldsymbol{x})-f_{\hat{y}}(\boldsymbol{x})\right]^2} \end{aligned} \tag{3.8}$$

Let us represent, $f_{\tilde{y}}(x) - f_i(x)$ as Δ_i and set of Δ_i over all classes be Δ. Hence, $\Delta = \{\Delta_i\}_{i\neq\hat{y}}$ and $\bar{\Delta}$ denote the mean of the set Δ. The first term after summation in the denominator is represented as U_1

$$U_1 = \bar{\Delta} = \frac{1}{N-1}\sum_{i\neq\hat{y}}\Delta_i = \frac{1}{N-1}\sum_{i\neq\hat{y}}\left[f_{\hat{y}} - f_i\right] \tag{3.9}$$

The second term after summation in the denominator can be represented as U_2 and further expanded to understand the effect of Temperature Scaling:

$$
\begin{aligned}
U_2 &= \frac{1}{N-1}\sum_{i\neq\hat{y}}\Delta_i^2 && \text{by } \Delta_i = f_{\hat{y}} - f_i \\
&= \frac{1}{N-1}\sum_{i\neq\hat{y}}\left(\Delta_i - \bar{\Delta} + \bar{\Delta}\right)^2 \\
&= \frac{1}{N-1}\sum_{i\neq\bar{y}}\left[\left(\Delta_i - \bar{\Delta}\right)^2 - 2\left(\Delta_i - \bar{\Delta}\right)\bar{\Delta} + \bar{\Delta}^2\right] \\
&= \underbrace{\frac{1}{N-1}\sum_{i\neq\bar{y}}\left[\Delta_i - \bar{\Delta}\right]^2}_{\text{Variance}^2(\Delta)} - \underbrace{\frac{2\bar{\Delta}}{N-1}\sum_{i\neq j}\left(\Delta_i - \bar{\Delta}\right)}_{=0} + \underbrace{\bar{\Delta}^2}_{\text{Mean}^2(\Delta)}
\end{aligned}
\tag{3.10}
$$

From Equation **??**, we can observe that U_1 measures How much the dominant value of the output of the neural network deviates from the remaining outputs, and from Equation **??** measures the variance of all the remaining outputs.

Referring back to Equation **??**

$$
\begin{aligned}
S_y(x;T) &= \frac{1}{N - \frac{1}{T}\sum_{i-1}^{N}\left[f_i(x) - f_i(x)\right] + \frac{1}{2T^2}\sum_{i-1}^{N}\left[f_i(x) - f_{\hat{y}}(x)\right]^2} \\
\mathrm{U}_1 &= \frac{1}{N-1}\sum_{i\neq\hat{y}}\left[f_{\hat{y}}(x) - f_i(x)\right] \\
\mathrm{U}_2 &= \frac{1}{N-1}\sum_{i\neq\hat{y}}\left[f_{\hat{y}}(x) - f_i(x)\right]^2 \\
S_y(x;T) &= \frac{1}{N\left(1 - \frac{1}{(N-1)T}\sum_{i\neq\hat{y}}\left[f_{\hat{y}}(x) - f_i(x)\right] + \frac{1}{2(N-1)T^2}\sum_{i+\hat{y}}\left[f_{\hat{y}}(x) - f_i(x)\right]^2\right)} \\
&= \frac{1}{N\left(1 - \frac{1}{T}\left(U_1 - \frac{U_2}{2T}\right)\right)}
\end{aligned}
\tag{3.11}
$$

From the Equation **??**, we can conclude that $S \propto \left(U_1 - U_2/2T\right)/T$. The component U_1 results in producing larger softmax scores for in-distribution images than out-of-distribution images. But U_2 has inverse-proportion effect on softmax scores $S \propto -U_2$. Choosing a large temperature reduces the negative effects of $U_2/2T$ resulting in in- and out-of-distribution samples being more distinguishable.

Input Preprocessing

The authors also proposed to perform input pre-processing by subtracting a small perturbation noise to the input image. This approach is inspired by the work done by **?**] in which a small perturbation

is added to make the softmax scores less effective for classification purposes thereby forcing the neural network to make an incorrect prediction.

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \text{esign}\left(-\nabla_{\boldsymbol{x}} \log S_{\hat{y}}(\boldsymbol{x}; T)\right) \tag{3.12}$$

$$\log S_{\tilde{y}}(\tilde{\boldsymbol{x}}; T) = \log S_{\hat{y}}(\boldsymbol{x}; T) + \varepsilon \left\|\nabla_{\boldsymbol{x}} \log S_{\hat{y}}(\boldsymbol{x}; T)\right\|_1 + o(\varepsilon) \tag{3.13}$$

where ϵ is the perturbation magnitude and $S_{\hat{y}}(\boldsymbol{x}; T)$ is the maximum temperature-scaled softmax scores produced by the network fed with an image $\boldsymbol{x}$. The authors had modified the input increasing softmax scores. This effect can be seen more in case in-distribution samples than out-of-distribution samples.

The behavior of softmax score in case of input perturbation can be understood by analyzing the log-softmax function as shown in Equation **??** with the input as perturbed image $\tilde{x}$. From extensive experimentation, the authors observed that the in-distribution samples tend to have a larger norm of gradients in comparison to out-of-distribution samples. In case of equal softmax values for in and out of distribution samples, But magnitude of gradients L_1 norm calculated using $|\nabla_{\boldsymbol{x}} \log S_{\hat{y}}(\boldsymbol{x}; T)\|_1$ is high for in-distribution sample.

3.3.3 Mahalanobis distance-based OOD Detector

?] proposed Mahalanobis distance-based out-of-distribution detector which assumes that the intermediate layer features of pre-trained models following class-conditional Gaussian distributions with tied covariances for all class distributions, and the classification confidence score for a new input $\mathbf{x}$ is defined as the Mahalanobis distance from the closest class conditional distribution. Mahalanobis distance is calculated using the Equation

$$M(x) = \max_c - \left(f(x) - \hat{\mu}_c\right)^T \hat{\Sigma}^{-1} \left(f(x) - \hat{\mu}_c\right), where.$$

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_c=c} f\left(x_i\right) \tag{3.14}$$

$$\hat{\Sigma} = \frac{1}{N} \sum_c \sum_{i:y_c=c} \left(f\left(x_i\right) - \hat{\mu}_c\right) \left(f\left(x_i\right) - \hat{\mu}_c\right)^T$$

The authors assume that the class-priors follow categorical distributions $P(t = c) = \beta_c / \sum_{c'} \beta_{c'}$ and class-conditional distributions can be modelled as Gaussian distributions with tied covariance $\mathcal{N}\left(\mathbf{x} \mid \mu_c, \Sigma\right)$.

The Posterior can be defined as follows using Bayes theorem

$$P(y = c \mid x) = \frac{P(x \mid y = c)P(x)}{P(x)}$$

As already mentioned above that likelihood is a Gaussian distribution

$$P(x \mid y = c) = \frac{1}{\sqrt{2\pi\Sigma}} e \frac{-\left(x - \mu_c\right)^T \Sigma^{-1} \left(x - \mu_c\right)}{2}$$

$$P(y = c \mid x) = \frac{\frac{1}{\sqrt{2\pi\Sigma}} e \frac{-(x-\mu_c)^\top \Sigma^{-1}(x-\mu_c)}{2} (\beta_c)}{\sum_{c'} \frac{1}{\sqrt{2\pi\Sigma}} e \frac{-(x-\mu_c')^\top \Sigma^{-1}(x-\mu_{c'})}{2}}$$

Applying logarithm on both sides

$$\log(P(y = c \mid x)) = \log\left(e \frac{-\left(x - \mu_c\right)^\top \Sigma^{-1} \left(x - \mu_c\right)}{2} (\beta_c)\right) - \log\left(\sum_{c'} e \frac{-\left(x - \mu_c\right)^\top \Sigma^{-1} \left(x - \mu_c\right)}{2}\right)$$

For simplicity purposes, let us represent $\left(\sum_{c'} e \frac{-(x-\mu_c)^\top \Sigma^{-1}(x-\mu_c)}{2}\right)$ as A

$$\log(P(y = c \mid x)) = \log\left(e \frac{-\left(x - \mu_c\right)^\top \Sigma^{-1} \left(x - \mu_c\right)}{2} (\beta_c)\right) - \log A$$

$$P(y = c \mid x) = -\frac{1}{2} \left(x - \mu_C\right)^\top \Sigma^{-1} \left(x - \mu_C\right) + \ln\left(B_C\right) - \log(A)$$

$$= -\frac{1}{2} (\underbrace{x^\top \Sigma^{-1} x}_{\text{constant}} - x^\top \Sigma^{-1} \mu_c - \mu_c^\top \Sigma^{-1} x + \mu_c^\top \Sigma^{-1} \mu_c) + \ln\left(\beta_c\right) - \log(A)$$

$$= -\frac{1}{2} \left(-2x^\top \Sigma^{-1} \mu_c + \mu_c^\top \Sigma^{-1} \mu_c\right) + \ln\left(\beta_c\right) - \log(A)$$

$$= x^\top \Sigma^{-1} \mu_c - \frac{1}{2} \mu_c^\top \Sigma^{-1} \mu_c + \ln\left(\beta_c\right) - \log(A)$$

$$P(y = c \mid x) = \frac{e^{\left(x^\top \Sigma^{-1} \mu_c - \frac{1}{2} \mu_c^\top \Sigma^{-1} \mu_c + \ln(\beta_c)\right)}}{\sum_{c'} e^{\left(x^\top \Sigma^{-1} \mu_{c'} - \frac{1}{2} \mu_{c'} \Sigma^{-1} \mu_{c'} + \ln(\beta_c)\right)}} \tag{3.15}$$

From the Equation **??**, we can represent posterior distribution with a softmax classifier by setting weights as $\mu^T \Sigma^{-1}$ and biases as $-\frac{1}{2} \mu_c^\top \Sigma^{-1} \mu_c + \ln(\beta_c)$. From this observation, we can infer that features from the final layers of pre-trained neural classifier can be approximated as a class-conditional Gaussian distribution with tied covariances, and the Mahalanobis distances from class means can be used as confidence scores.

The authors also proposed an input processing similar to the one proposed in ODIN by [**?**]. The preprocessing is done as per the following Equation

$$\tilde{x} = x - \varepsilon \operatorname{sign}\left(-\nabla_x M(x)\right) \tag{3.16}$$

Where $M(x)$ is the Mahalanobis distance calculated using Equation **??** and ϵ represents the perturbation

magnitude.

3.4 Datasets

The majority of the well-performing deep learning-based object detection models are trained in a supervised manner. Hence, the dataset should consist of RGB camera images and 2D bounding box labels. The use of a standard dataset as a benchmark is of importance, as it allows to draw a standard comparison between different algorithms and to set targets for solutions.

PASCAL VOC [**?**], is one of the widely used datasets for benchmarking of object detection purposes introduced in 2005 with further iterations in 2007 and 2012. The motivation of introducing the PASCAL VOC dataset is *"methods are now achieving such good performance that they have effectively saturated on these datasets"*. The current version of PASCAL VOC introduced in 2012 consists of 27,450 objects in 11.530 images segregated into 20 independent classes. Microsoft Common Object in Context (COCO) introduced in 2015 is another dataset with labels consisting of bounding box labels, instance segmentation labels, image captions, and key points. It consists of 2.5 million object labels in 382,000 images. But, PASCAL VOC and COCO datasets mainly consist of images with objects as the center of the image and the objects are the dominant contents of the image.

As the research in Autonomous Driving (AD) bloomed there were many real-world automotive datasets like KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) [**?**], Berkley Deep Drive dataset [**?**], Cityscapes [**?**], ApolloScape [**?**], nuScenes [**?**], and Waymo open Dataset [**?**]. The datasets vary from each other based on the types of sensors, recorded location, year, variation of illumination, and weather conditions. KITTI dataset introduced by **?**] was recorded in 2011 by the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago Object Detections in the city of Karlsruhe, Germany. The dataset is collected using 2 RGB and grayscale cameras, a Velodyne HDL-64E, Global Positioning Systems (GPS), and gyroscope sensors. The dataset consists of 15k images with a resolution of 1242x375 and its respective annotation. The dataset is collected from different environments: Residential area, On-road, and College campus. It contains labels for 2D and 3D bounding boxes for cars, vans, trucks, pedestrians, person-sitting, cyclist, tram, and miscellaneous objects as well as semantic and instance segmentation labels. Cityscapes is another pioneering work done by many collaborators across Europe, with the task of pixel-wise and instance-level segmentation under consideration. The dataset consists of RGB images collected using a 2MP stereo camera with a rolling shutter, vehicle odometry, and GPS coordinates. There are a total of 25K images with a resolution of 2048x1024 along with depth data calculated from the stereo-camera setup. There are a total of 30 different classes considered during annotation belonging to various superclasses like vehicles, persons, ambient, and traffic objects.

Since the above-mentioned datasets are of small scale and don't contain variation in the environmental conditions, researchers in Berkley proposed a new crowd-sourced dataset called BDD100K from **?**] to solve the task of object detection and instance segmentation. This dataset consists of over 100,000 RGB images of 1280x720 resolution and is collected in Newyork, Berkley, San Francisco, and Bay area. Data labels along with task-specific labels also include weather condition, time of the day and consists of 9

different classes: Bus, traffic-light, traffic-sign, pedestrian, motorcycle, truck, car, train, and bicycle. To scale-up the amount of data available for training purposes new datasets like Waymo Open Dataset proposed by **?**], nuScenes from **?**], Lyft level-5 dataset introduced by **?**] are released. These datasets like KITTI **?**] are multi-modal. The sensor suite consists of RGB cameras, LiDARs, GPS, and Gyroscopes for various measurements. nuScenes also include 5 RADARs. These datasets present a well-calibrated 3D point cloud from LiDAR, RGB images, and RADAR point cloud in the case of nuScenes dataset. These datasets cover various conditions with their size scaling in the order of terabytes.

Recent works of research had focused on evaluating the robustness of the object detection models. But, all the proposed datasets mentioned previously are captured in well-maintained and structured environments like structured traffic, road lanes, similarity in the background, and objects appearance. To address the lack of novelty in background and object variance researchers proposed Indian Driving Dataset **?**], which is collected in unstructured traffic and traffic conditions. The dataset consists of 46.5K images with 15 different classes annotated. The dataset is collected in and around the cities of Bangalore and Hyderabad in India. Canadian Adverse Driving Dataset (CADC) **?**] is a dataset that aims to promote the exploration of autonomous driving solutions in severe snowy or adverse weather conditions. The dataset features 56K images and a 7K LiDAR point cloud with 3D bounding box annotations belonging to 10 different classes. The dataset is collected by researchers from Toronto Robotics and AI Laboratory (TRAIL) and Waterloo Intelligent Systems Engineering Lab (WISE Lab).

Table 3.1: Comparison of various autonomous driving datasets available with image data

Name	Recording area	Dataset size (# images)	Categories	Weathers / Lightning conditions
A2D2 [**?**]	Germany	40K	14	Clear, Cloudy, Rainy, and Sunny / Day
A*3D [**?**]	Singapore	39K	7	Clear, Rain / All conditions in day
EuroCity Persons [**?**]	Various countries in Europe	47K	1	Clear, Rain, Snow, Cloudy / All conditions in day
Waymo Open [**?**]	NA	200K	3	Clear, Rain, Snow, Cloudy / All conditions in day

Lyft Level 5 [?]	NA	55K	9	Clear, Rain, Snow, Cloudy / All conditions in day
Argoverse [?]	Pittsburgh, Pennsylvania, Miami, Florida	113K	15	Clear, Rain, Cloudy / All conditions in day
PandaSet [?]	San Francisco, El Camino Real	60K	28	Clear / All conditions in day
nuScenes [?]	Boston, Singapore	1.4M	23	Clear, Rain / All conditions in day
ApolloScape [?]	Multiple places in China	144K	24	Clear / All conditions in day
KITTI [?]	Karlsruhe	7.4K	8	Clear / Day
IDD [?]	Bangalore, Hyderabad	10K	15	Clear / Day
BDD100K [?]	Berkley	70K	10	Clear, Rain, Cloudy / All conditions in day
CityScapes [?]	50 cities in Germany	5K	30	Clear, Rain, Cloudy / Day
KAIST [?]	Seoul	7.5K	1	Clear / All conditions in day
CADC [?]	Toronto	56K	10	Clear, Rain, Cloudy, Snow / Day

4

Benchmarking

This chapter describes the detection datasets that are used to benchmark the current Out Of Distribution (OOD) methods given an In Distribution (ID) datasets, Metrics used to define the benchmark limits and the expected behavior from the OOD detector.

As mentioned in Section **??** and shown in Figure **??**, the OOD dataset should include images with considerable semantic shift either in the background of the image or in the objects present in the image. To establish a benchmark, we set the following requirements from our OOD detector which are adapted from the work done by **?**].

The OOD detector should be able to:

1. Reject the image samples from a different domain. For example, images collected in a different country with very different road conditions from the image data used for training the object detection model.

2. Reject low-quality images. For example, images with low contrast, improper lighting, motion blur, etc.

3. Reject objects in the image which are unseen during the training.

4.1 Datasets

Proposed frameworks to detect OOD samples used one dataset as ID and another dataset which is semantically very different from OOD. Adapting such a setting is not possible in the case of object detection, as the datasets available tend to be similar to others: the majority of the agents in the dataset tend to be similar. This section outlines the datasets utilized in this work, including the exploratory data analysis and the choices made to choose the training dataset.

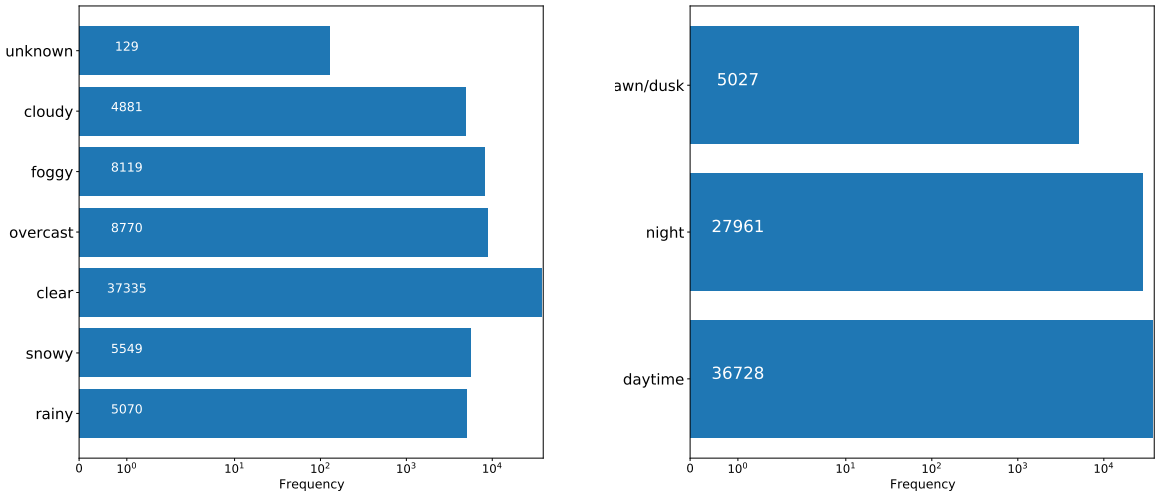
4.1.1 Berkley Deep Drive (BDD100K) Dataset

Berkley Deep Drive dataset which consists of 100,000 images, Hence this dataset is coined as BDD100K by **?**]. The collected images are annotated with object bounding boxes, road and object segmentation, lane-level markings, instance segmentation labels. The dataset is particularly wholesome as it covers

various weather conditions (rainy, snowy, clear, overcast, partly cloudy, and foggy) as well as the various times in a day (daytime, night, dawn/dusk). This variance in the environment and illumination condition has made BDD100K a good and natural representation of a driving domain.

The dataset consists of 1.8 million bounding boxes with objects belonging to 13 different classes ('pedestrian', 'rider', 'car', 'truck', 'bus', 'train', 'motorcycle', 'bicycle', 'traffic light', 'traffic sign', 'other vehicle','other person', 'trailer'). The distribution of Object-instances, average area of each Class-instance, Weather, and Time of Day are shown in Figure **??**, Figure **??**, Figure **??**, and Figure **??** respectively. Hence, we chose to use BDD100K dataset as In-Distribution dataset.

Figure 4.1: Example images from the BDD100K datasets introduced by **?**]



(a) Weather count distribution

(b) Time of day distribution

Figure 4.2: Weather and Illumination variations in BDD100K dataset

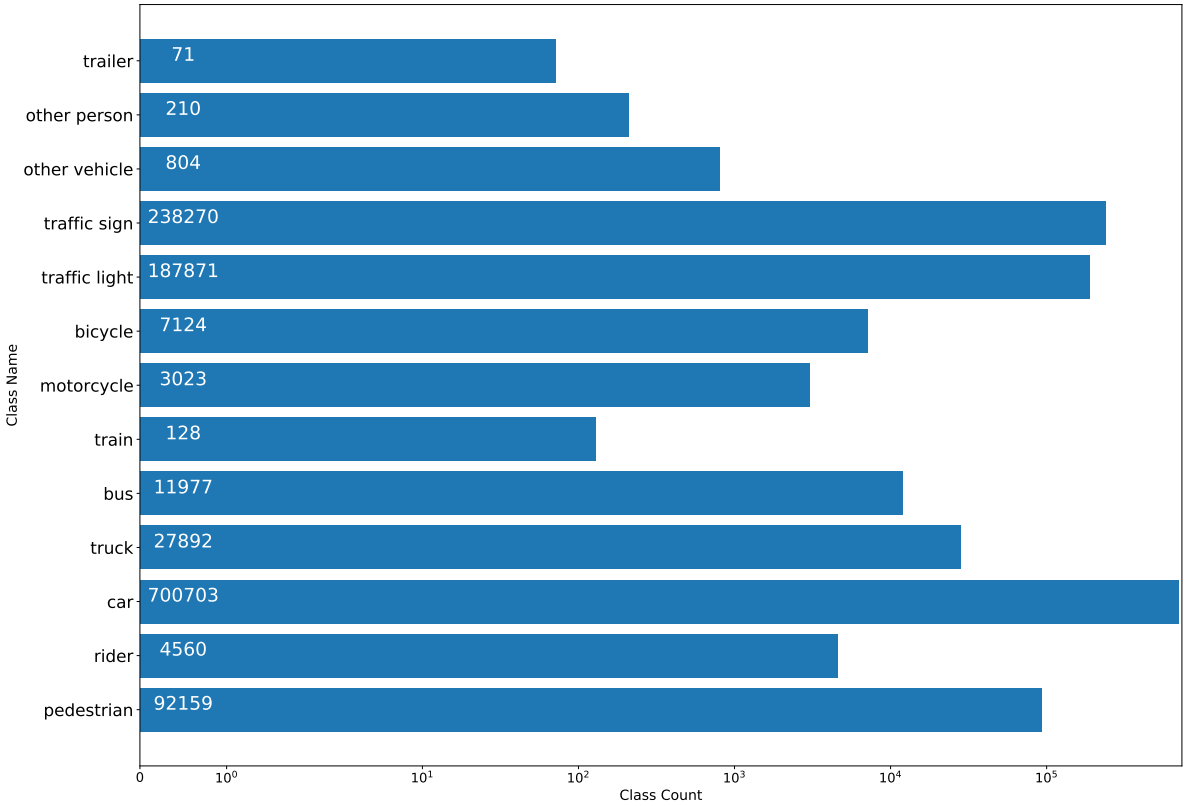


Figure 4.3: Number of object instances for all the 13 classes present in BDD100K dataset with x-axis representing the count of object instances and y-axis representing the name of the object instance.

4.1.2 Indian Driving Dataset

Indian Driving Dataset is collected from very less structured environments around cities like Hyderabad, Bangalore in India. The road scenes differ highly from the environment with less structured traffic participants and a large variance in object types appears in the dataset. Figure **??** shows few random samples from the IDD dataset.

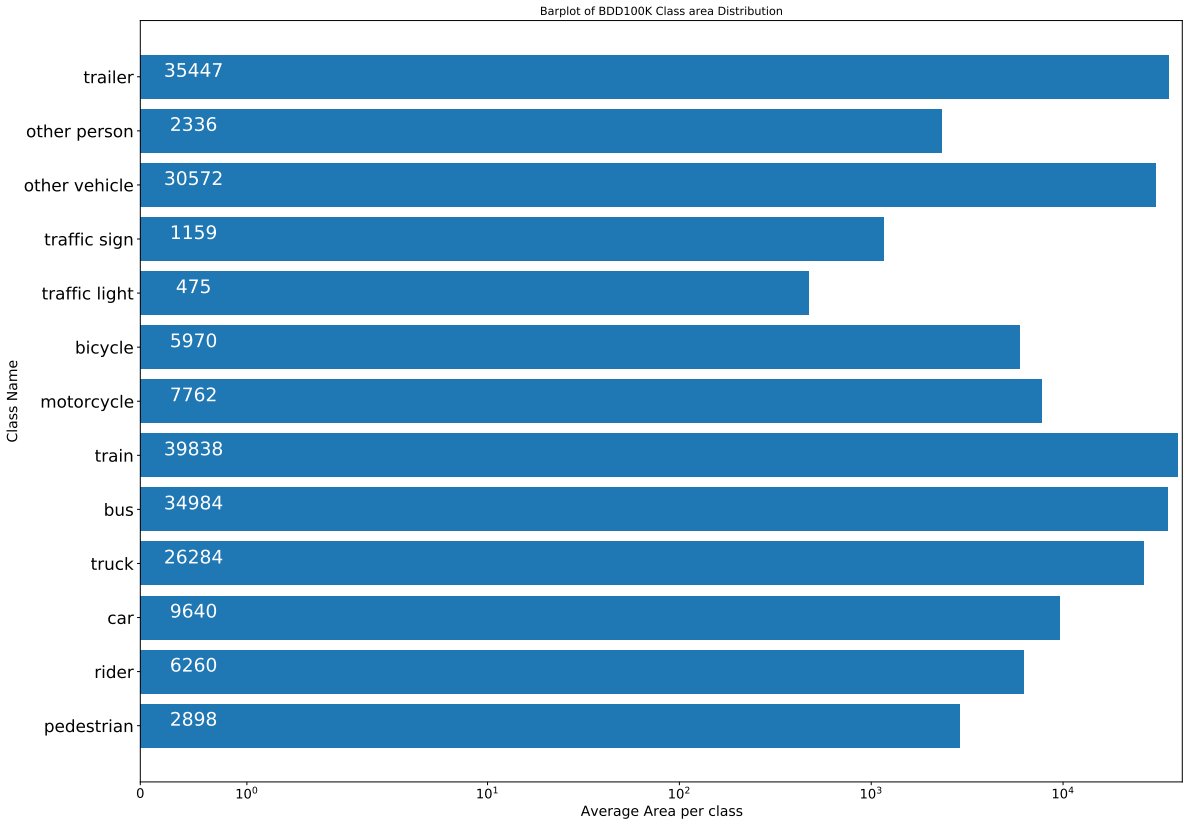


Figure 4.4: Plot representing average bounding box area per class

Figure 4.5: Example images from the IDD dataset introduced by **?**]

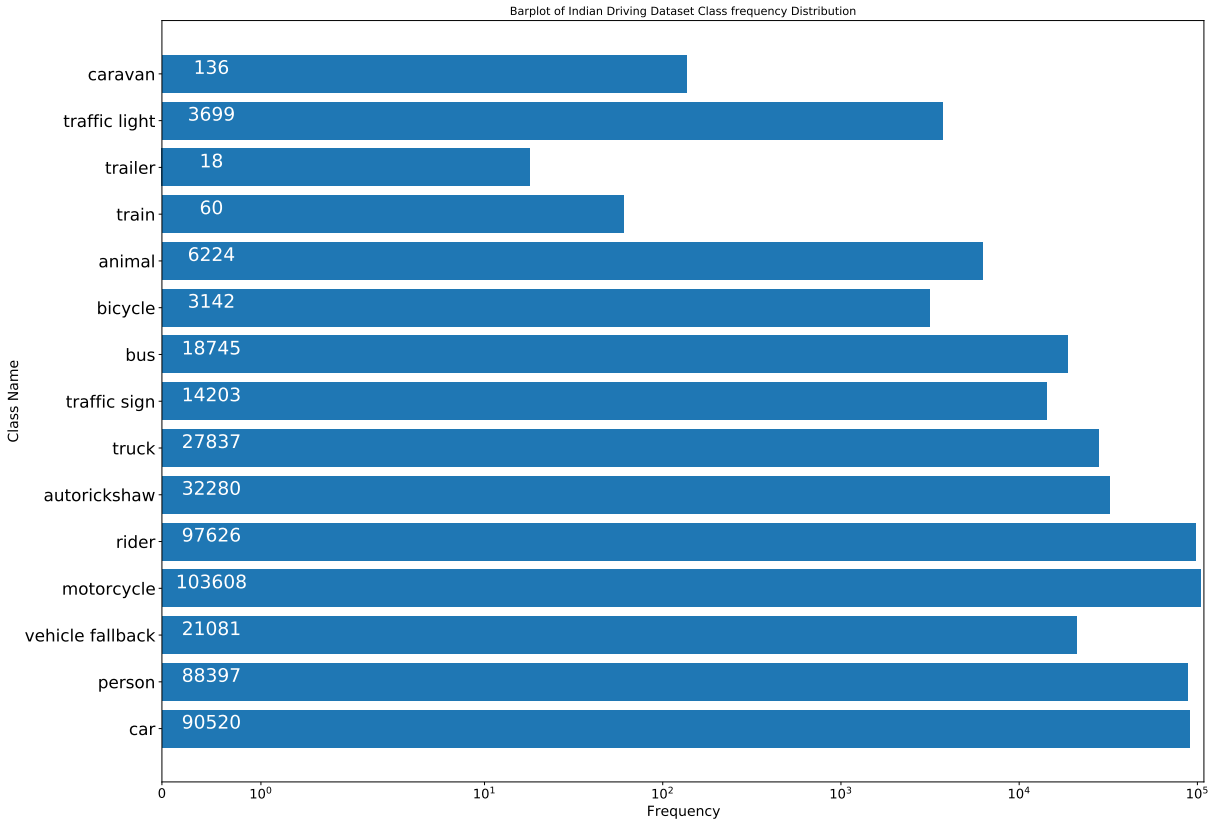


Figure 4.6: Number of object instances for all the classes present in IDD dataset with x-axis representing the count of object instances and y-axis representing the name of the object instance.

The reasons for using IDD as OOD dataset are

1. The Figures **??** and **??** show that the road and driving conditions in which IDD dataset is recorded is very different from BDD100K dataset. Thus making IDD dataset an out-of-domain dataset.

2. The intra-class variance among the classes in IDD dataset is very high. For example the OOD detector should assign a high novelty score to a car in Figure **??** compared to cars in Figure **??** - **??**.

(a) i (b) ii (c) iii (d) iv

Figure 4.7: Intra-class variance in car class from IDD dataset

3. The appearance of few classes differ between BDD100K and IDD datasets. For example in IDD the appearance of Truck class highly varies from BDD100K. From the Figure **??**, truck samples in Figure **??** are to be detected with low novelty score as they are similar to truck classes in BDD100K. But, the trucks in Figure **??** are to be detected with high novelty score.

(a) (b)

Figure 4.8: Inter-dataset variation in case of Truck class between BDD100K show in Image **??** and IDD shown in Image **??**

4. Presence of novel-class like Auto-Rickshaw, The samples present in Figure **??** to **??** are to be detected with high novelty scores as they are not present in BDD100K dataset.

(a) i (b) ii (c) iii

Figure 4.9: Novel class of Auto-Rickshaw present in IDD dataset

4.1.3 BDD100K-Weather Dataset

Though IDD dataset can be used for evaluating the OOD detection method in case of semantic shift in the object of interest to the object detector. But evaluating only on IDD dataset does not cater to the use case when the background of the images is affected either due to internal camera conditions or external weather conditions. To address this issue, we used ClimateGAN proposed by **?**], to visualize the potential consequences caused due to the climate changes such as wildfires, flooding, and dense smog. To generate photo-realistic images with disasters of the authentic scenes. To create these photo-realistic images authors harnessed Generative Adversarial Networks (GAN) proposed by **?**] using two networks: a generator and a discriminator.

In case of flooding, the GAN model leverages both real-world images and simulated data from a virtual city modeled using Unity3D. Using these images, the generator model extracts the location of water in an image in case of flooding. Then a discriminator model based on GauGAN to create the water textures using the input image. The ClimateGAN model is trained using a custom dataset called the Mila-Simulated-Flood dataset proposed by the authors which consist of 5540 non-flooded scenes to train the generator and 1200 images of flooded scenes to train the discriminator network.

The pipeline for creating photorealistic flood images is shown in Figure **??**, an input x is processed using an Encoder E. The decoders D, S, and M process z the output of Encoder E. D produces a depth map d, S produces a segmentation map s, and M generates a binary flood mask by sequentially normalizing z using various SPADE [**?**] blocks. The painter model P is also based on SPADE blocks which generate an image of a photo-realistic flood based on the input image x and binary predicted mask m. The visualization of the image at every step is shown in Figure **??**. The authors also generated realistic images with wild-fire and dense smog

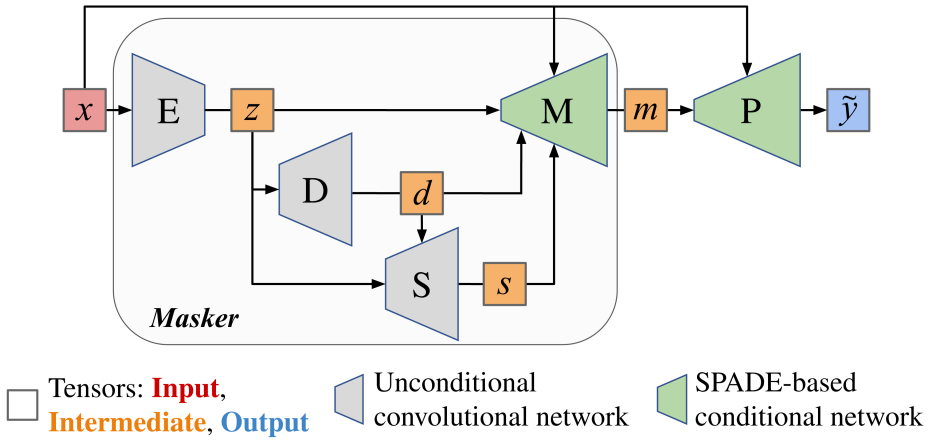


Figure 4.10: The generator model used for creating images with various climates. The generator is mainly based on SPADE blocks [**?**].

To create a dataset for bench-marking we used images from the validation dataset of BDD100K

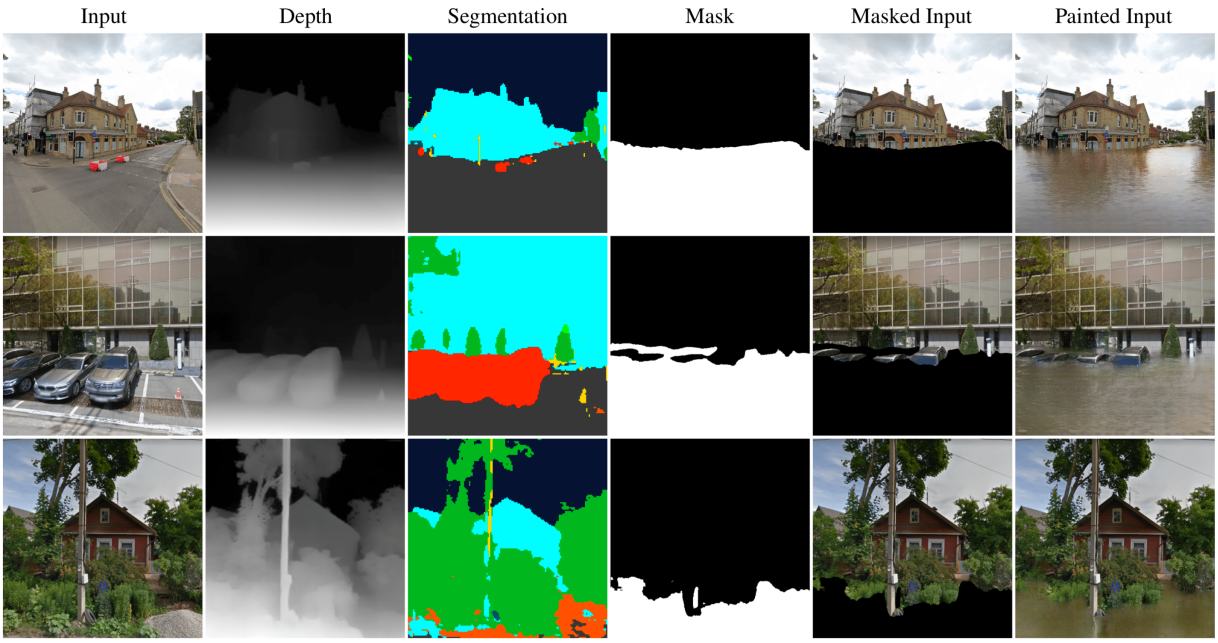


Figure 4.11: Pipeline for creating photorelistic images with flood proposed by **?** , p.6]. Reusing the segmentation map for generating the mask of the road and the depth map for calculating the flatness of the ground

to create a dataset of images with the flood, wild-fire, and dense-smog. Combining the IDD and the ClimateGAN model generated images we generated a dataset and from further on the dataset is referred to as OD^2 (Out-of-Distribution detection for Object Detection).

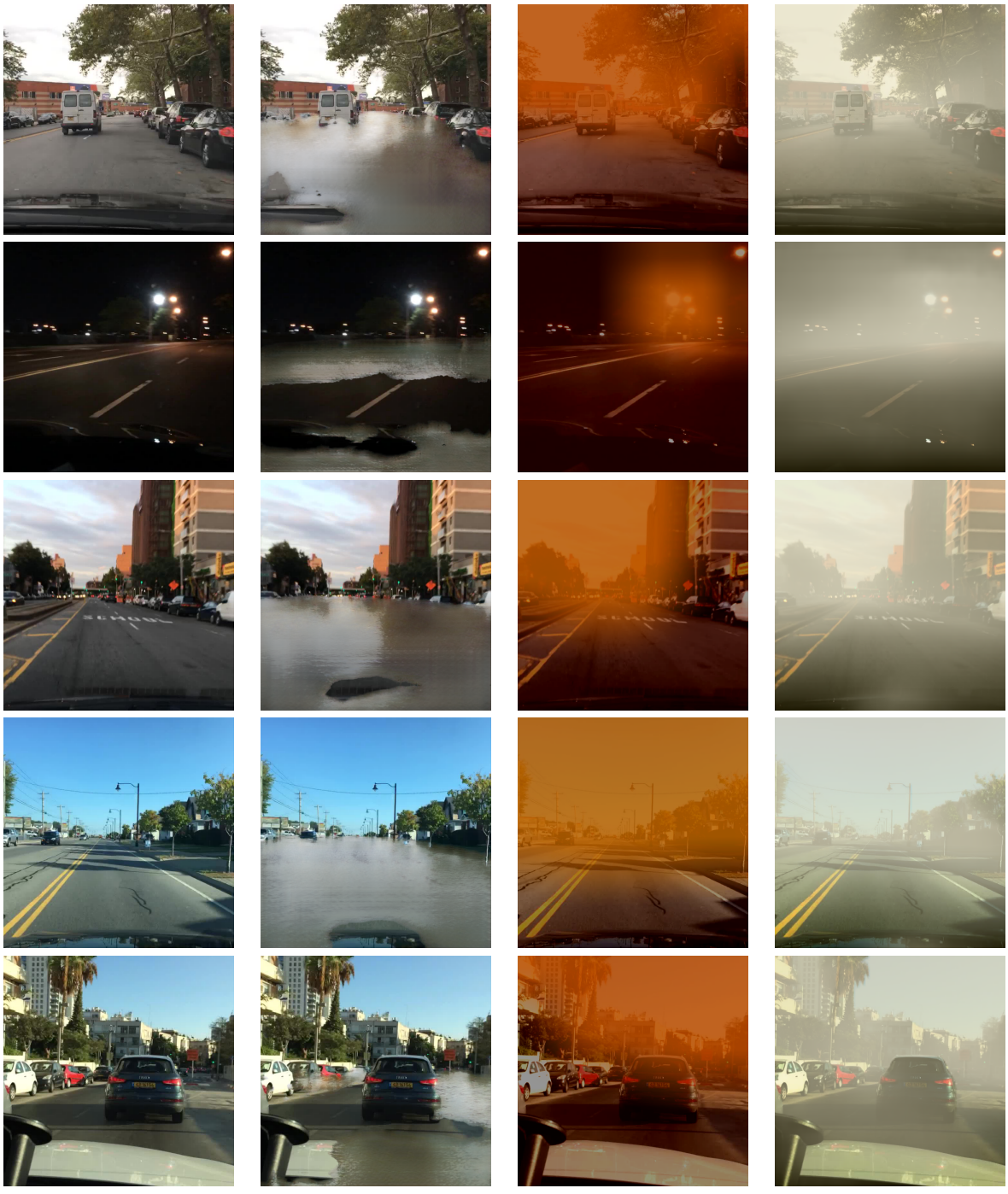


Figure 4.12: Samples from the dataset generated using ClimateGAN model and the images from the validation split of the BDD100K dataset

4.1.4 Performance Metrics

In this section, we present various possible evaluation metrics that help in measuring the prediction quality when faced with OOD samples.

Area Under Receiver Operating characteristic Curve

AUROC is a measurement used to evaluate the performance of the classification problem at different thresholds. ROC is a probability curve, that is plotted between True Positive Rate (TPR) against False Positive Rate (FPR) and AUROC represents the measure of the classifiers ability to distinguish between the classes and can be used to summarize the ROC curve. The higher the AUROC value is the better is the classifier at the task under consideration.

This can be is used to evaluate the OOD detector that is posed as a classification problem as per the Equation **??**. To pose the OOD detection as a classification problem,

- Assign a label of **0** to In Distribution (ID) sample.
- Assign a label of **1** to OOD sample.
- Use the novelty score that is provided by the OOD detector to perform the classification between ID and OOD samples.

The TPR and FPR values from the classification problem can be used to calculate the AUROC value that can represent the ability of the OOD detector in classifying and ID samples.

Probability and Entropy

Entropy quantifies the uncertainty in the classification head of the network. Entropy is calculated using the Equation **??** from the mean of the softmax scores $s_{\mathbf{x}^*}$ as in Equation **??** from each detection.

For an input x^*, the entropy and probability of a detection belonging to class c among a total count of C classes is calculated using Equations **??** and **??**.

$$p\left(c \mid \mathbf{x}^*\right) = \frac{1}{N} \sum_{i=1}^{N} s_{\mathbf{x}^*}^i \tag{4.1}$$

$$\text{Entropy} = -\sum_{i=1}^{C} P\left(c_i \mid \mathbf{x}^*; \mathcal{D}\right) \ln P\left(c_i \mid \mathbf{x}^*; \mathcal{D}\right) \tag{4.2}$$

Box Deviation

Box deviation quantifies the uncertainty in bounding box regression. It represents how much each box $\hat{\mathbf{v}}_{\mathbf{x}^*}^i$ varies from their mean value.

1. Final bounding box detections are calculated as the mean value of the detections regressed during N forward passes.

2. Covariance matrix of the detections is calculated using the following the equation

$$C\left(\mathbf{x}^{*}\right)=\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{v}}_{\mathbf{x}^{*}}^{i}\hat{\mathbf{v}}_{\mathbf{x}^{*}}^{i^{T}}-\mathbf{I}_{\mathbf{x}^{*}}\mathbf{I}_{\mathbf{x}^{*}}^{T} \tag{4.3}$$

3. Box deviation is the square root of the trace of the covariance matrix $C(x^{*})$.

The box deviations span from $[0,+\infty)$, the higher the value represents the higher the epistemic uncertainty in regressed bounding box parameters.

4.2 Out-of-Distribution for Object Detection (OD^2) Dataset Summary

In this section, we summarize the key features of the proposed OD^2 dataset along with the method used to compare various OOD methods for object detection.

To test the performance of OOD detector on images:

1. In case of a low-quality image, we propose the usage of images with various weather conditions like partly cloudy, foggy, overcast, etc. To confirm that all types of objects are present in the images, we also propose the usage of smog-filled images generated using Climate-GAN.

2. From a different domain, we propose the usage of images with fire and flood overlay produced using Climate-GAN along with images from IDD dataset.

3. with novel classes, we propose the usage of images from IDD dataset with Auto-Rickshaws.

Task Description

This section outlines the tasks we are planning to achieve using OD^2 dataset

Object Detector Performance This task ensures the object detector's performance on the closed dataset that is chosen for training purposes. In OD^2 dataset this is achieved using the test split of the BDD100K dataset.

Detector Robustness This task is similar to task 1. Except, the data that is used for inference is from a different but plausible weather condition.

OOD Detection This task is to detect the samples which are from a known data distribution but are semantically different from the data used for training.

Multi-class Novelty Detection This task is to quantify the ability of OOD detector to produce high novelty scores for unknown-unknown samples. The class of Auto-Rickshaw in IDD dataset helps in evaluating this task.

Out-of-Domain Detection This task is to test the ability of the OOD detection methods to provide high novelty scores when the input sample is from a different domain to the training dataset. The Flood and WildFire datasets generated can be used to serve this purpose.

The OD^2 dataset can be summarised as shown in Table **??**

Table 4.1: Table showing various type of images to address the OOD cases presented in Section **??**. The table also show the sources of the images that are collected from along with the expected behavior of the novelty score for each OOD class of dataset

Purpose	**Dataset Source**	**Classes**	**Novelty Score Behavior**	Task
In-Distribution	BDD100K	Pedestrian, Rider, Car, Truck, Bus, Motorcycle, Bicycle, Traffic sign	Low Novelty score	Object detector performance
Low light and bad image quality	BDD100K (non-clear weather) and Climate-GAN generated Smog images	Pedestrian, Rider, Car, Truck, Bus, Motorcycle, Bicycle, Traffic sign	Medium Novelty Score	Detector Robustness
Classes with semantic-variance	IDD	Trucks, Motorcycles, Traffic Sign	High Novelty Score	OOD detection
Novel Classes	IDD	Auto-Rickshaws	High Novelty Score	Multi class novelty detection
Out-of-Domain images	Climate-GAN generated Flood and Fire images	Pedestrian, Rider, Car, Truck, Bus, Motorcycle, Bicycle, Traffic sign	High Novelty Score	Out-Of-Domain detection

5

Experimental Setup

This chapter describes the various selection choices and preparations done to perform the experimentation are presented. We discuss the chosen object detector model, the framework that is used to model and train the model, hyper-parameter tuning, and the selected OOD detection algorithms that are adapted to the object detection task. We also detail various metrics that are used to distinguish between OOD and ID detections.

5.1 Object Detection Model

In this work we selected SSD [**?**] object detection model. The working of SSD is explained in detail in Section **??**. We used a pre-trained SSD model from Perception for Autonomous Systems (PAZ) software library introduced by **?**].

5.1.1 Perception for Autonomous Systems

PAZ is a hierarchical library with multiple Application Programming Interfaces (APIs) levels that is proposed to solve various perception problems in robotics domain. It allows for the modification of software pipelines at various hierarchical level based on requirement. Though PAZ is similar to previously proposed libraries like TensorFlow Object Detection API (TF-OD) [**?**] and Detectron 2 [**?**], the ease of modification to the object detection pipeline in order to incorporate the OOD methods make PAZ our library of choice.

PAZ has three levels of abstractions, a high-level abstraction referred to as a Pipeline, consists of application-ready function for object detection, data-augmentation, and input pre-processing, a mid-level abstraction referred to as a Processor. Processors can be used to perform small computations that are needed to provide solutions to various applications or entirely novel algorithms. Hence, introducing flexibility and re-usability of various implemented functions in PAZ, a low-level abstraction referred to as a Backend. Backend is meant to be used and modified without any effect on high-level or mid-level functionalities. These backend functions can also be accessed directly without any need for the usage of either Processors or Pipelines. PAZ mainly depends on Tensorflow 2.0 [**?**], OpenCV [**?**] and Numpy [**?**]. In order to ease the implementation and visualizing of the results we also used Matplotlib [**?**], sci-kit learn [**?**] and seaborn [**?**].

5.1.2 Tensorflow

Tensorflow introduced by **?**] is a library that is initially developed as an internal research tool at Google and later released as open-source software. It is used for the manipulation of tensors that is required in various scientific fields that need intensive numerical calculations. There are multiple APIs provided by Tensorflow to make the machine or deep learning models deployable in various devices with CPUs, GPUs, and TPUs. Tensorflow at its current iteration of version 2. × .× is built using keras [**?**] as its backend, thereby providing a more hierarchical approach. The integration of keras into Tensorflow enables the implementation and building of neural networks as a graph, or a sequence of configurable modules of neural network layers, loss functions, activation functions along with various regularization and weight-initialization techniques. Tensorflow also provides various special utility functions called Callback functions. These Callback functions are event-triggered and can be used for various tasks like learning rate schedule, early stopping, and logging of various intermediate values.

5.1.3 Compute Platform

Training any DNNs is a computationally intensive task that involves high-precision multiplication of input images, extracted features, and weights in the order of millions. To optimize the computations, these tasks are run on GPUs. Using GPUs would result in the reduction of training time to hours instead of weeks. In this work, we used the High-Performance Cluster (HPC) available at DFKI, Bremen. The HPC contains a total of 16 compute nodes (12 data storage, 3 GPU-compute, and a login node). Each node consists of two octa-core Xeon E5-2630 v3 (Haswell) aided by a 128GB ECC RAM. Among the three GPUs nodes, two of those nodes are powered by five NVIDIA Titan-XP each with a processing memory of 12GB, and one node has four NVIDIA Tesla V100 cards with 32GB RAM. The resource management and job scheduling are done using Slurm.

5.1.4 Training

In this work, for the training of SSD object detection model, we used the hyper-parameter setting defined in the original work on SSD by **?**]. These hyper-parameter settings include the number of anchor boxes, size of anchor boxes, batch size, filter sizes of all the NNs layers, loss functions and threshold values for confident scores, IoU overlap scores for non-maximum suppression. We also used the data augmentation techniques that are implemented in the original work. These techniques include random image cropping, random image mirroring, random image expansion, and photometric distortions like modifications in images brightness, contrast, hue, and saturation.

In the original work, the SSD network is trained using e ”multi-step” learning rate policy in which the learning rate is decreased after a certain number of epochs. The learning rate is decayed using a pre-set decay rate as 0.1, this decay rate is generally referred to as ***gamma***. To speed up the learning process, we reduced the batch size to a value of 8 and also chose to use Stochastic Gradient Descent (SGD) proposed by **?**] with a momentum of 0.9.

The network is trained for a total of 80 epochs following the above setting and a callback is employed to save all model parameters values to the disk for every 5 epochs. To benchmark the performance of the SSD object detection network on BDD100K dataset, we decided to use the leader board from the object detection competition from the 2018 Workshop of Autonomous Driving. This competition is based on BDD100K dataset and the results are not publicly available. But, the runner-ups presented their work titled *CFENet: An Accurate and Efficient Single-Shot Object Detector for Autonomous Driving* [**?**] in which the scores are mentioned. Hence, we decided to use the scores mentioned in their work for bench-marking purposes. According to this work, the winner and runner-up of the competition scored an mAP value of 0.331 and 0.297 respectively.

5.2 Out Of Distribution Detection

This section describes the implementation of different Out-Of-Distribution (OOD) detection techniques based on the SSD300 object detection model. The methods are originally proposed to solve the task of object classification but are adapted to detect the OOD samples in the area of object detection.

5.2.1 ODIN

In ODIN, the input is perturbed with noise as shown in Equation **??**. To perform ODIN, two forward passes are needed to perturb the input image. In the first pass, a gradient of the loss with respect to the input image is calculated and is scaled using a perturbation magnitude. The input image is modified by subtracting the perturbation calculated in the previous step. This results in an addition of two hyper-parameter Temperature (T) and Perturbation magnitude (ϵ). These hyperparameters are tuned using a fraction of test images sampled from IDD dataset. To calculate the gradient we used *tf.GradientTape*, which records all the differentiation operations information, and a sub-method *tf.gradient* calculates the gradient value using the operations that are recorded in the context of the defined tape.

5.2.2 Mahalanobis Distance based OOD Detector

In this method, the training images are used to calculate class-wise mean vectors of each feature and a tied covariance matrix of all the classes using the features from the penultimate layer. The calculation has to be done according to Section **??**. The class-specific mean vectors and the covariance matrix are used to calculate the Mahalanobis distance of a test sample from each known class. The Mahalanobis distance to the nearest class can be used as a novelty score.

To calculate the mean of class-specific feature vectors and the covariance matrix, we used images with other classes masked out by the mean of the images as shown in Figure **??**. This is done to extract exact class-dependent features.

The penultimate layer of the SSD that is used in this work is *concatenate-2* which is of shape (78588 $\times$ 1). Hence, the covariance matrix is of shape (78588 $\times$ 78588), and calculating it is not possible with the available computational resource present. Hence, we decided not to pursue this as a possible option for an OOD detector in the case of object detection.

(a) Perturbation Magnitude=0.005, Temperature=10

(b) Perturbation Magnitude=0.005, Temperature=1000

(c) Perturbation Magnitude=0.05, Temperature=10

(d) Perturbation Magnitude=0.05, Temperature=1000

(e) Perturbation Magnitude=0.2, Temperature=10

(f) Perturbation Magnitude=0.2, Temperature=1000

Figure 5.1: Perturbed images with different Perturbation Magnitudes and Temperatures. Observe the increase in the magnitude of noise added to the image with an increase in magnitude and temperature values

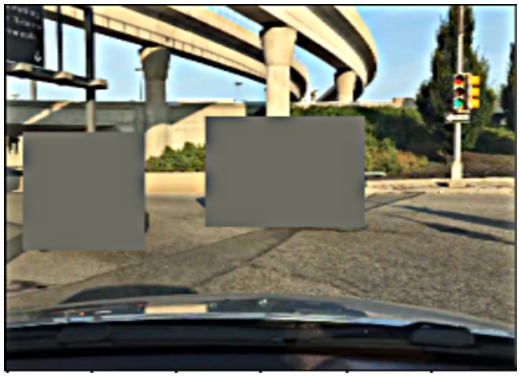


Figure 5.2: Class specific image with all classes masked except Traffic sign

5.2.3 Bayesian-SSD Object Detector

Bayesian-SSD object detector is used to quantify uncertainty in SSD object detection model. As mentioned in Section **??**, the convolution filters are modeled as a Gaussian distribution parameterized by mean and variance. To model a Bayesian-SSD we used the Tensorflow-Probability library introduced by **?**], we use Flipout version of the Convolution layer introduced by **?**] from the Tensorflow-Probability library to model the Bayesian Layers in the SSD network.

But previous works [**?** **?**] done using the Flipout layers to model the neural network led to the model being under-fitted when trained. We believe the reason for underfitting the model is the inherent stochasticity in weight sampling. To avoid this issue, we decided to only model the layers highlighted in green from the Image **??**. We adapted the weight initialization proposed by **?**] and initialized the priors with a mean value of **0** and a standard deviation of **1**. The Bayesian-SSD300 model is trained similarly to the SSD300 model explained in Section **??**.

To perform uncertainty quantification, we performed **20** forward passes through the Bayesian-SSD300 model. Since the weights are randomly sampled from the trained prior distributions for every forward pass, each forward pass produces a different output thereby providing an ability to calculate the uncertainty of the SSD300 model in its detections.

5.2.4 Sub-Ensemble based SSD Object Detector

In Sub-Ensembles, a model is divided into a trunk network **T** and a task network **K**. In this work, we chose the layers highlighted with green color in Image **??** to be selected as task layers. The rest of the layers in the model are chosen to be trunk network. As already mentioned in Section **??** the trunk network is fixed and will not be trained. Hence, we decided to restore the weights of the trunk network from the best checkpoint of the SSD300 model.

For training the task network, we initialize all the layers in the task network randomly and are re-trained. Since the initialization of weights is random, training the layers results in a task network

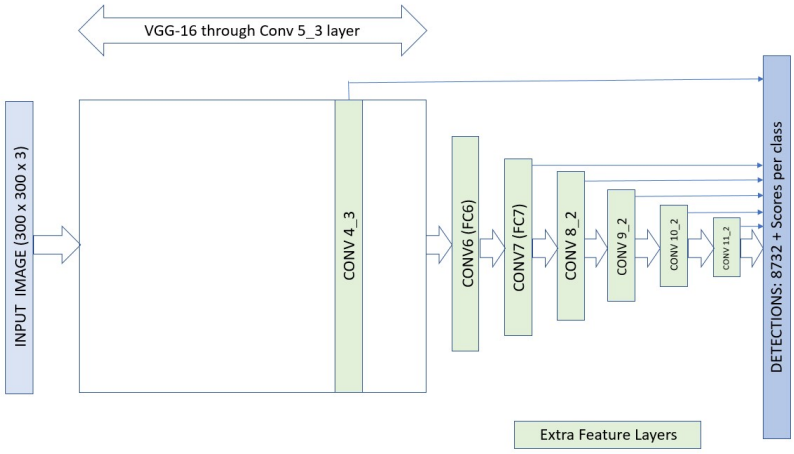


Figure 5.3: SSD300 model with highlighted layers that are modelled as Bayesian layers

having different weights. Hence, every model produces different output for a similar input, this sort of behavior helps us in quantifying the uncertainty in the detections made by the SSD300 model.

To maintain the experiment similarity, we chose to use 20 different task networks to form a sub-ensemble model. Each of the 20 task networks regresses different boxes and the final detection made is the mean of the detections. The different detections made from each task network can be used to produce various uncertainty metrics defined in Section **??**.

6

Experiments

In this chapter, we present various experiments performed with the proposed methodologies for object detection and Out-Of-Distribution (OOD) detection.

- In Experiment 1 [Section **??**], an SSD300 object detection model is trained and validated on BDD100K dataset with default boxes from VOC dataset. The default boxes are tuned to BDD100K dataset and this data specific tuning had resulted in a significant improvement in the object detection model performance.
- In Experiment 2 [Section **??**], we investigate the ability of softmax-scores and ODIN based OOD detector to detect OOD samples and evaluate them.
- In Experiment 3 [Section **??**], we implement Bayesian Neural Networks (BNN) and Sub-Ensembles for uncertainty quantification.
- In Experiment 4 [Section **??**], we test the ability of uncertainty based OOD detection using BNNs and Sub-Ensemble models.
- In Experiment 5 [Section **??**], we compare uncertainties in various weather conditions and analyze the results by comparing entropy and bounding-box deviation.

6.1 Training and Evaluating an SSD Object Detector using BDD100K Dataset

The main aim of this experiment is to train the SSD object detection network with the hyper-parameter setting as mentioned in Section **??**. In order to train the SSD network, we used BDD100K object detection dataset, more details about the dataset are discussed in Section **??**. Though the dataset that is open-sourced consists of 100,000 images. But, 20,000 of those images do not have ground-truth annotations provided for evaluation purposes by the authors. Hence, we decided to use 60,000 of the 80,000 images from the training dataset for training the SSD network. From the rest of the 20,000 images, we used 10,000 images for validation purposes and the other 10,000 images are isolated for testing and benchmarking purposes.

Table 6.1: Average Precision values of each class in BDD100K dataset using SSD300 model with and without tuned prior boxes

	Agents								
Models	**Pedestrian**	**Rider**	**Car**	**Truck**	**Bus**	**Motorcycle**	**Bicycle**	**Traffic Sign**	**mean**
SSD300	0.006	0.004	0.095	0.083	0.15	0.045	0.092	0.001	0.059
SSD300 - Tuned	0.165	0.135	0.479	0.389	0.389	0.163	0.213	0.186	0.265

The choice of default boxes as well as matching between prior boxes and the ground truths is explained in Section **??**. Figure **??** shows the image with matched prior boxes that are fed to the network for training.

Results

The Average Precision of all the predicted classes from object detection on the test dataset with 10,000 images are presented in Table **??** and the related Precision-Recall (PR) curves are shown in Figure **??**.

Observations

- Prior boxes that are tuned for the VOC dataset are not suitable for harder object detection datasets like BDD100K or KITTI.
- This poor performance of SSD is attributed to the lack of positively matched ground-truth boxes to the prior boxes resulting in providing poor quality prior boxes for the training of the object detector.

Tuning the Prior boxes

The above experiment showed that the model's performance is poor because the prior boxes are not tuned correctly. Hence, the knowledge that is encoded in prior boxes scale and aspect ratios is important for the better performance of SSD object detection model. The following hyper-parameter changes are done in order to improve the SSD performance on BDD100K dataset:

- From the Figure **??** the majority of object that are present in the dataset are very small. Hence, the scale factors s_{min} and s_{max} are reduced so as to match the small object to the prior boxes. The post-tuning scale limits are set to 0.1 and 0.86 respectively.
- Adapting to the method proposed by **?**], scales for each prediction layer are set to: s_1=0.1, s_2=0.29, s_3=0.48, s_4=0.67, s_5 = 0.86. In addition to these scales, we also used a scale of s_0=0.05 for Conv4-3 layer inside the VGG-19 network this makes the object detection network to detect small objects.
- The scales are set to $a_r^{BDD100K} \in (0.5, 1, 1.5, 2, 2.5)$. This selection is done based on the scales of objects in BDD100K.

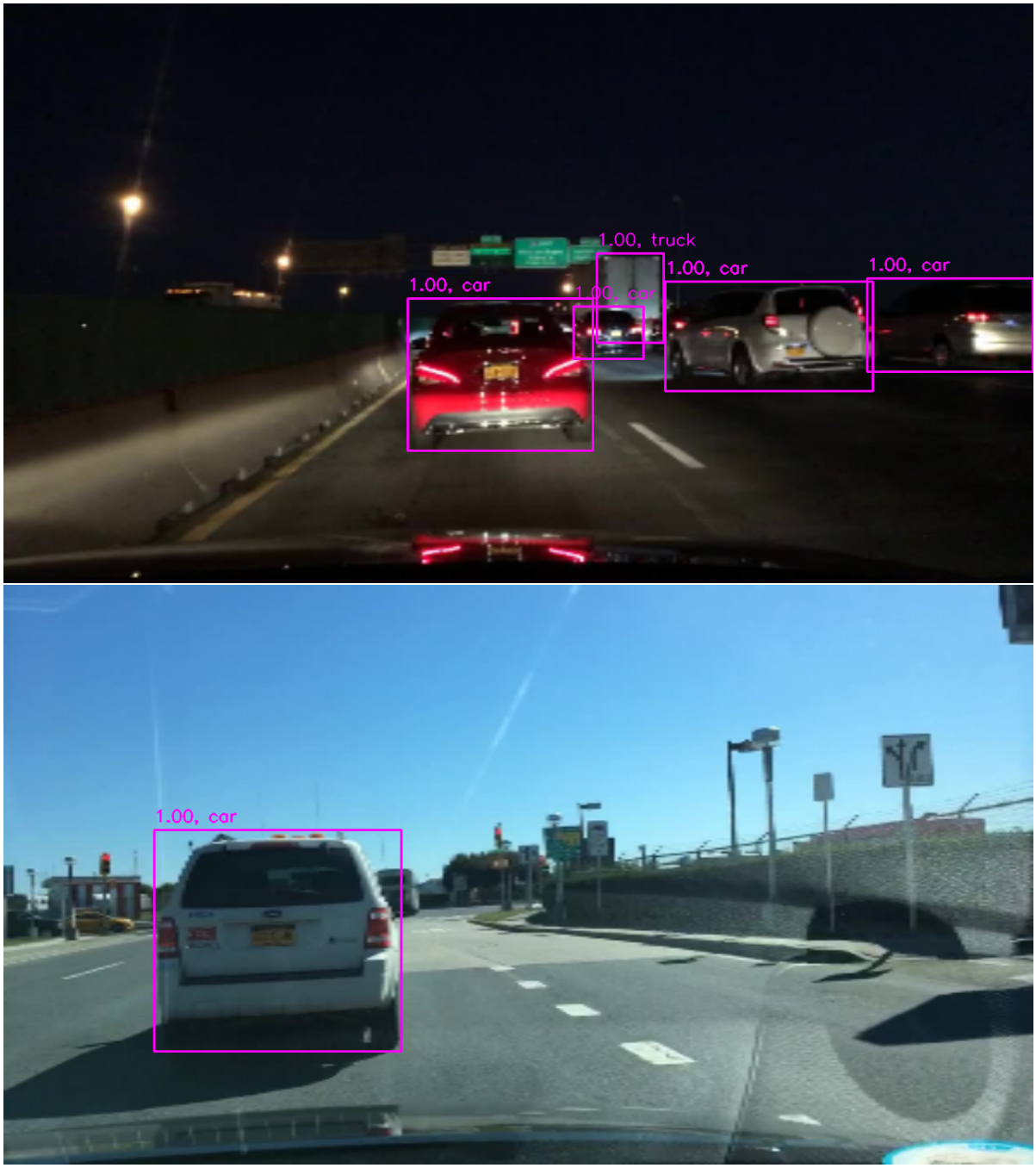
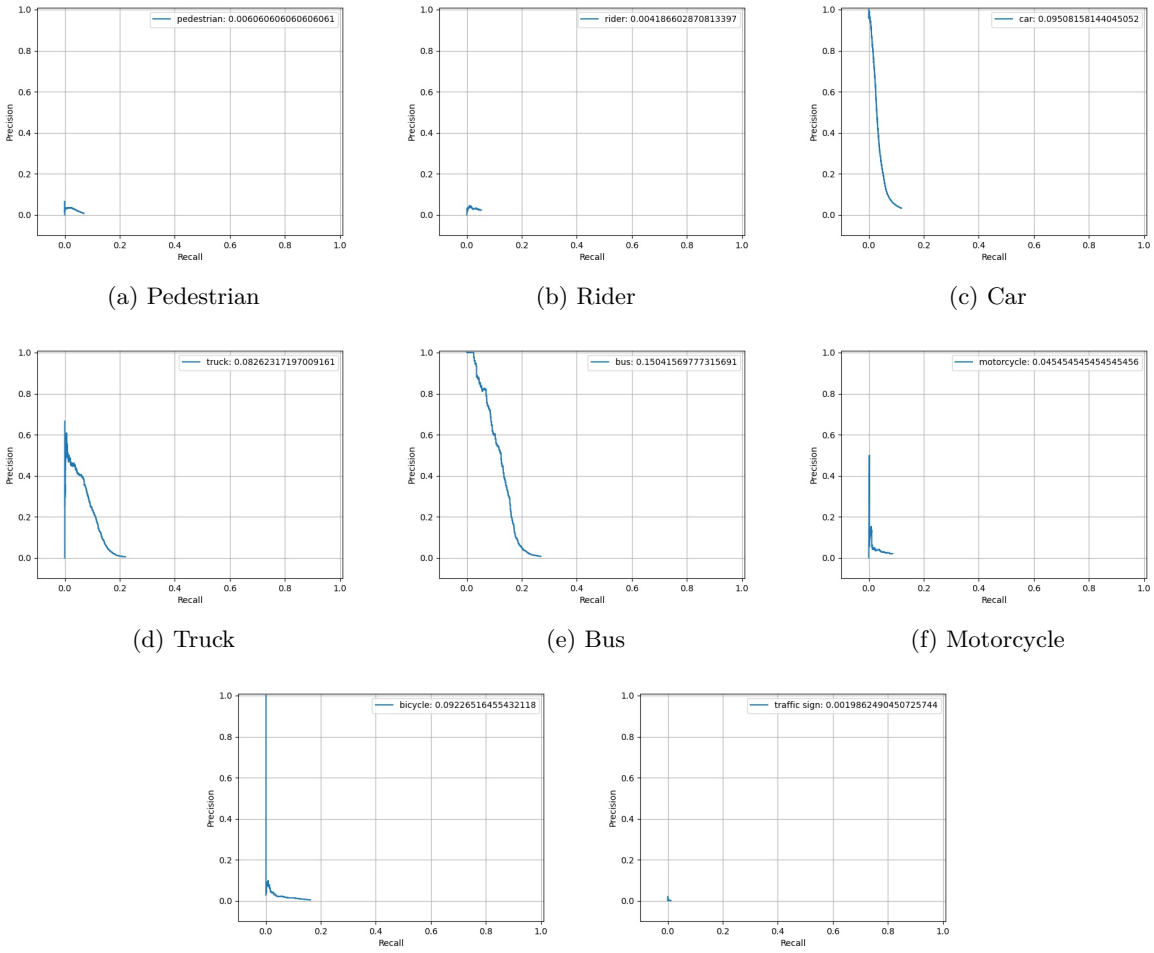


Figure 6.1: Image visualizing the matched ground truths to the prior boxes as per the matching strategy established in the SSD's original work by **?**]. We can observe that the small object boxes are not matched to the prior boxes resulting in low-quality training data provided.



(a) Pedestrian (b) Rider (c) Car

(d) Truck (e) Bus (f) Motorcycle

(g) Bicycle (h) Traffic Sign

Figure 6.2: Precision recall curves for each class in BDD100K dataset

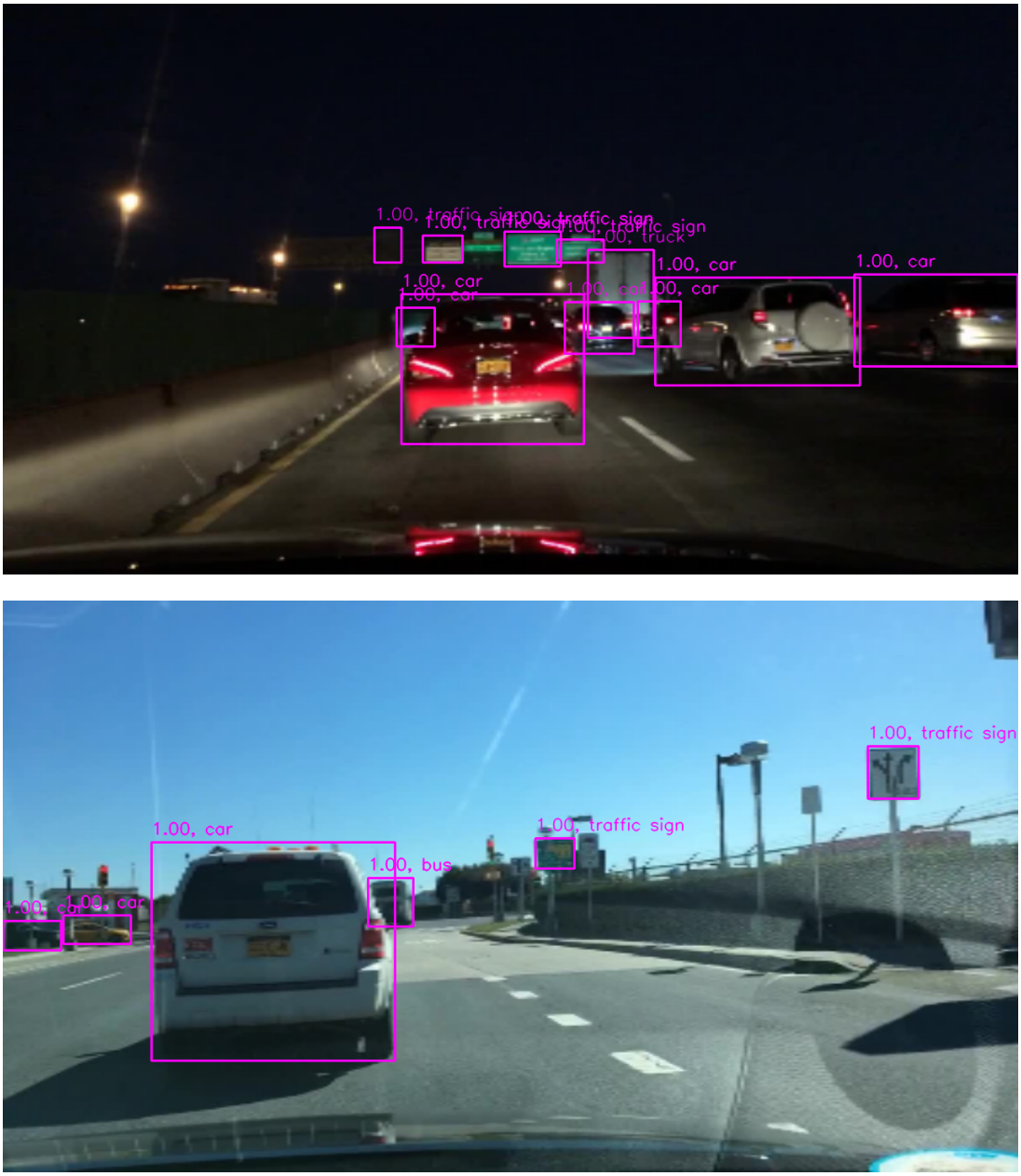
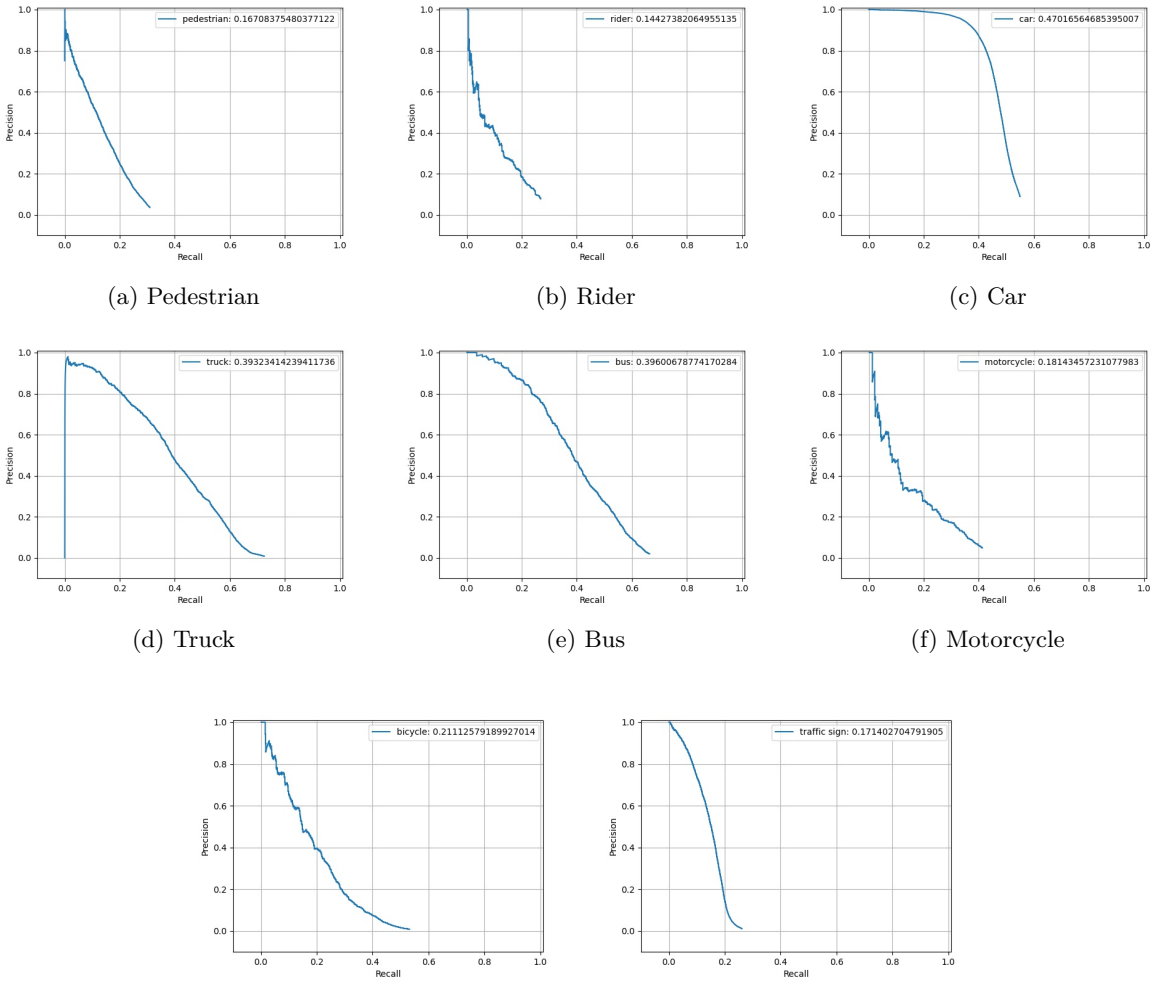


Figure 6.3: Image showing the matched ground truths to the prior boxes as per the matching strategy established in the SSD's original work by **?**] after tuning the prior boxes to encode BDD100K specific information. Comparing this Image with Image **??**, we can observe that there is huge increase in the positively matched ground truth boxes



(a) Pedestrian (b) Rider (c) Car

(d) Truck (e) Bus (f) Motorcycle

(g) Bicycle (h) Traffic Sign

Figure 6.4: Precision recall curves for each class in BDD100K dataset with prior boxes tuned using BDD100K dataset. We observed a huge increase on the performance of SSD300 model compared to performance shown in Figure **??**.

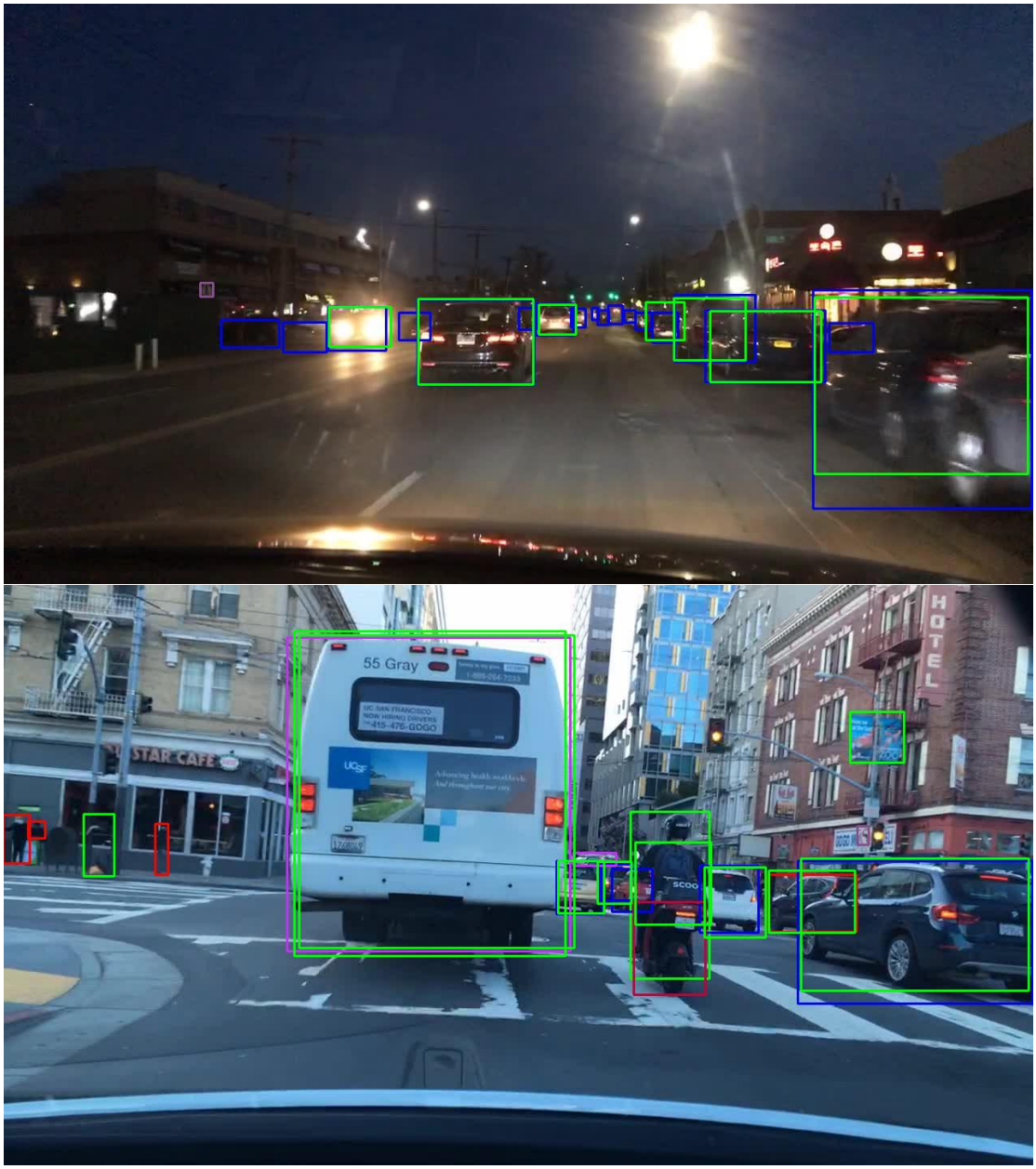


Figure 6.5: Detections made by SSD300 model on BDD100K dataset. The ground-truth is marked in various different colors other than green and the detected boxes are represented in green. The detector worked well at the night and also detected relatively small objects.

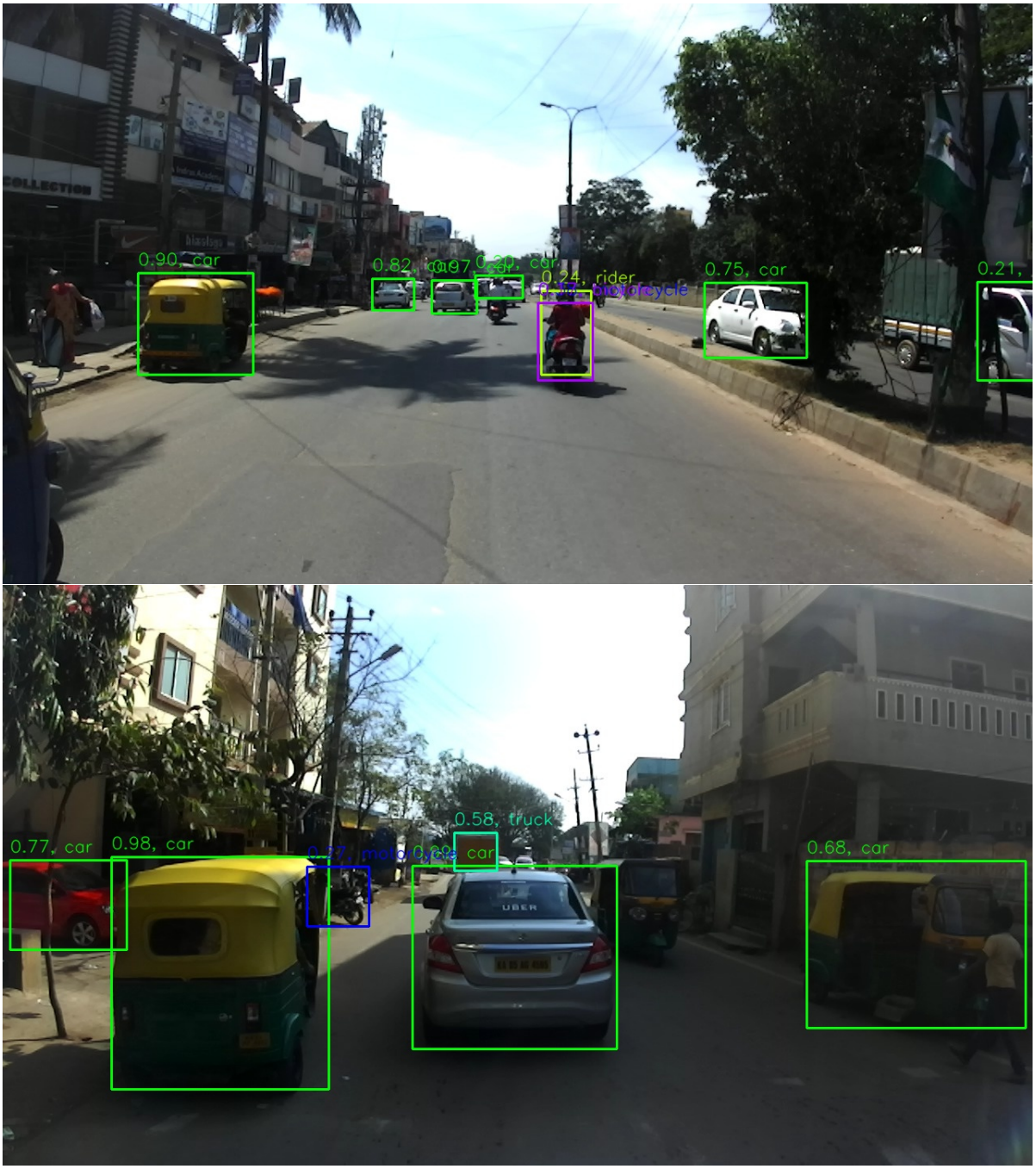


Figure 6.6: Detections made by SSD300 model on IDD dataset. We can observe that the OOD object (Auto-Rickshaw) present in the image is detected and is classified as car with a very high probability of greater than 0.9

Observations

- We observed an improvement in the performance of the object detection model with proper tuning of prior boxes.
- Current SSD object detector with truncated VGG-19 is comparable to one of the best performing models on the BDD100K object detection leader-board.
- The object detector model still struggle to detect small objects (Motorcycle, Bicycle, Traffic Sign, Pedestrian) compared to large objects (Car, Truck, Bus). On an average, the average precision values of large objects are 60% greater than smaller objects.

6.2 Out Of Distribution Detection

In this Section, we investigate various methods for performing OOD detection. The methods used are Max softmax [**?**], ODIN [**?**] explained in Section **??** explained in Section **??** and Uncertainty-based methods [**?** **?** **?**] using BNNs and Sub-Ensembles for uncertainty quantification. To perfrom this experiments, we use the OD^2 dataset that is proposed in the Section **??**. The performance of the OOD detector is measured using the metrics that are explained in Section **??**

6.2.1 Max Softmax

The main aim of this experiment is to test the ability of the softmax probability scores from the classification head of the object detector in classifying whether the detection is made from ID and OOD detection. The experiment is performed by posing the OOD detection problem as a classification problem as explained in Section **??**. To visualize the ability of max-softmax score to classify BDD100K(ID) and IDD(OOD), we use ROC curve as shown in Figure **??** and the Area Under the Receiver-Operating characteristic Curve can be used as a comparable metric to other methods.

Since this is currently used method to decide the models' confidence in its output. The AUROC score produced by this method will be used as a baseline to benchmark the ability of other methods. The results prove that the softmax probability score is not dependable to solve OOD problem. This inability of softmax scores can also be understood from the sample provided in Figure **??**, in which an OOD sample is assigned a high probability of greater than 0.9.

6.2.2 Softmax-based ODIN

The main aim of this experiment is to test the ability of ODIN in performing out-of-distribution detection in the case of complex datasets like BDD100K and IDD. According to the original work on ODIN, the authors showed that the softmax scores of ID samples are increased at the same time the softmax scores of OOD will have minimum effect on them. This increases the chances of classifying ID and OOD samples using the softmax scores after perturbation.

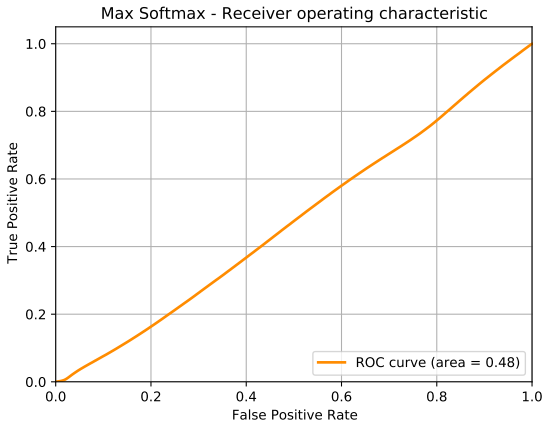


Figure 6.7: ROC curve representing the ability of softmax score in detecting whether the sample is In Distribution or Out Of Distribution

Hyperparameter Optimization

As previously mentioned this method introduces two more hyper-parameters, a perturbation magnitude ϵ and Temperature T. From the explanations in Section **??** we can understand that choosing the best values for these hyperparameters is very important to make a reliable OOD detection possible. Image **??** shows different perturbed images with different magnitudes and temperatures.

Since the original work on ODIN is done using very simple classification data the perturbation magnitude and temperature did not work for a complex object detection dataset. Hence, we decided to tune these parameters. To tune these parameters, we performed a grid search over a range of 100 input perturbation magnitude values ranging from 0 to 1, and temperature values are chosen from a set of values from 1 to 1000 at equal intervals are used. The grid search is performed to keep the same performance as with the non-perturbed image, the models' performance is judged by measuring the False Negative (FN) count from the detections. From the extensive search performed, the perturbation magnitude and temperature are set to 0.2 and 1000 respectively.

The ROC curve for classifying BDD100K (ID) and IDD (OOD) using the softmax scores after perturbations is visualized in Figure **??**.

Observations

- The AUROC score is improved by 12.5% with applying the perturbation and temperature scaling defined in ODIN compared to the baseline method.
- The original work proposed the usage of datasets that are semantically very different and have only one object per image.
- In our work, the image has multiple objects with varying sizes resulting in ODIN not being as effective as shown in the original work by the authors.

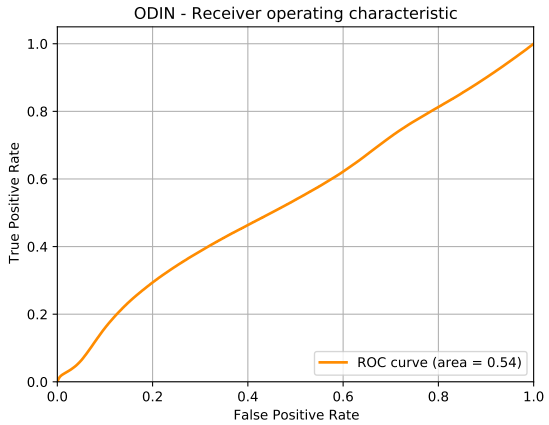


Figure 6.8: ROC curve representing the ability of softmax score after applying temperature scaling and perturbations in detecting whether the sample is In Distribution or Out Of Distribution. There is a slight increase in AUROC score, but is not significant to improve the safety of the perception systems.

- There is an improvement in OOD detection with the pre-processing of the image using perturbation magnitude and Temperature. But a further increase in the hyper-parameter values resulted in deterioration of object detector performance.

6.3 Uncertainty based Out Of Distribution Detection

This experiment is aimed at investigating the ability of uncertainty quantification methods using Bayesian Neural Networks and Sub-Ensembles in performing OOD samples in the images.

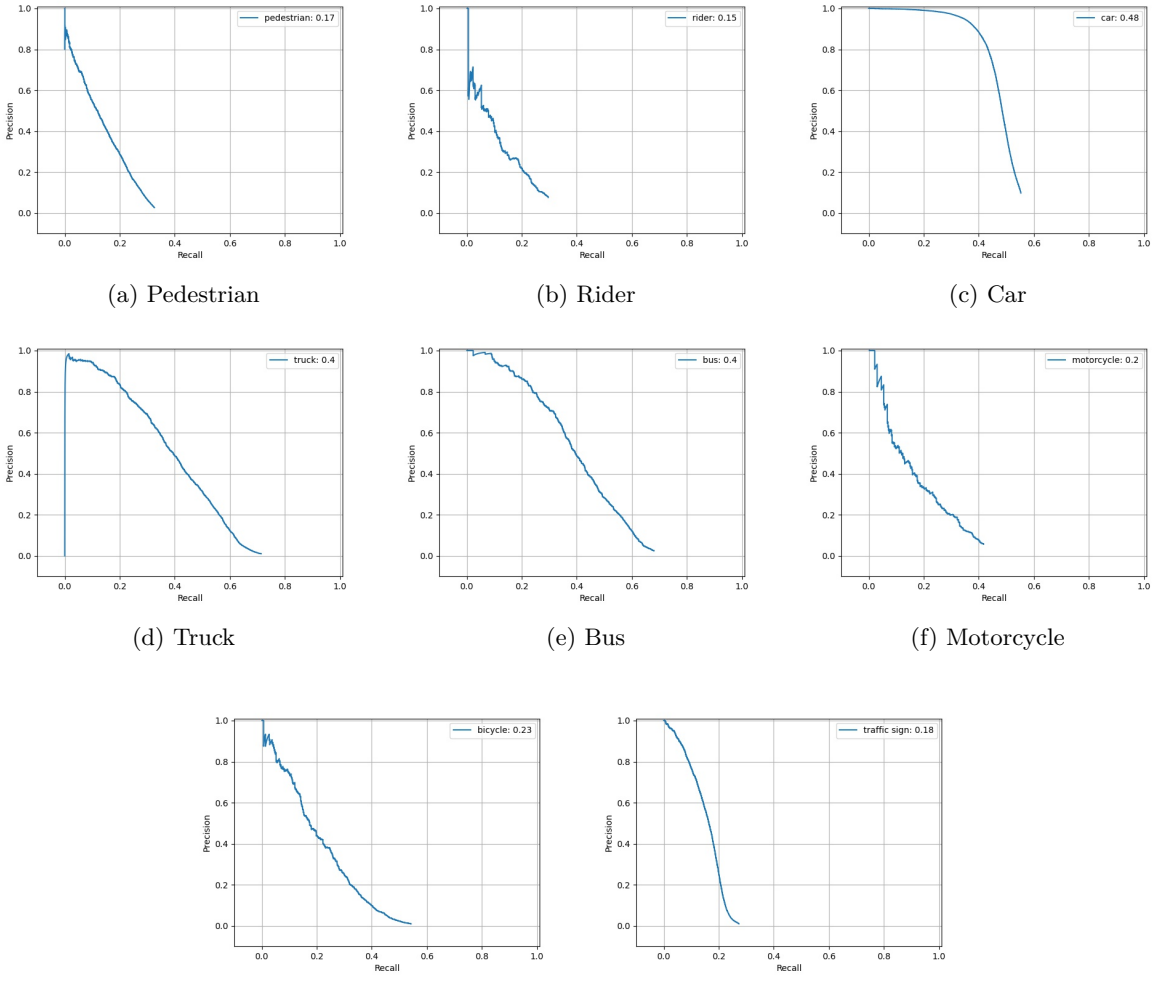
6.3.1 Training Bayesian and Sub-Ensemble based SSD Object Detector

First, we investigate the ability of BNNs and Sub-Ensemble-based SSD300 object detection model to perform object evaluation and compare their performances with the object detection model evaluated in Section **??**. From the precision-recall values and curves reported in Table **??** and Figures **??** and **??**, we observed that there is a 4.1% increase in mean average precision values using Bayesian-SSD300 model and a 0.7% increase in mean average precision values using SubEnsemble-SSD300 model compared to SSD300 model.

Hence, it can be confirmed that the Bayesian-SSD300 model outperformed the SSD300 and SubEnsemble-SSD300. These results are also corroborated by the work proposed by **?**] to solve the task of segmentation using Flipout sampling-based Bayesian layers, they proved that the bayesian version of their U-Net model had out-performed the normal and ensemble version of their models.

Table 6.2: Average Precision values of each class in BDD100K dataset using Bayesian-SSD300 and SubEnsemble-SSD300 models. We observed an increase in performance of the object detector models compared to values reported in Table **??**.

	Agents								
Models	**Pedestrian**	**Rider**	**Car**	**Truck**	**Bus**	**Motorcycle**	**Bicycle**	**Traffic Sign**	**Mean**
Bayesian - SSD300	0.172	0.149	0.476	0.4	0.401	0.196	0.232	0.184	0.276
SubEnsemble - SSD300	0.167	0.144	0.47	0.393	0.396	0.181	0.211	0.171	0.267

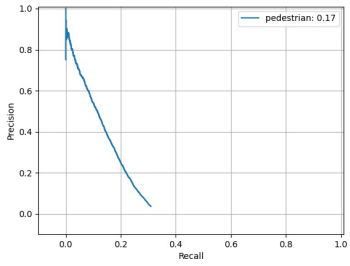


(a) Pedestrian (b) Rider (c) Car

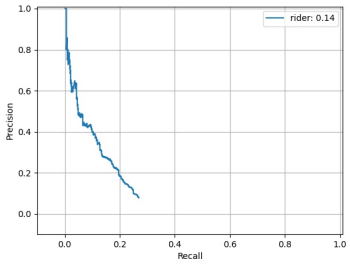
(d) Truck (e) Bus (f) Motorcycle

(g) Bicycle (h) Traffic Sign

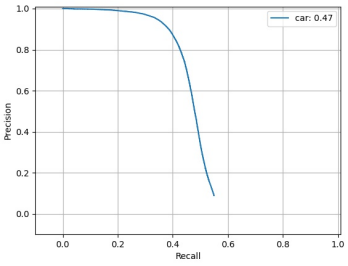
Figure 6.9: Precision recall curves for each class in BDD100K dataset using Bayesian-SSD300 model with prior boxes tuned using BDD100K dataset



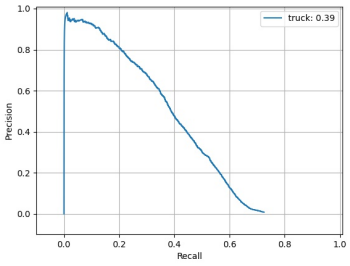
(a) Pedestrian



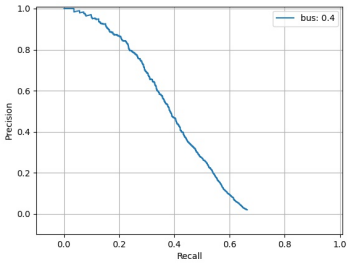
(b) Rider



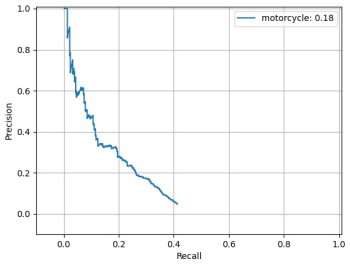
(c) Car



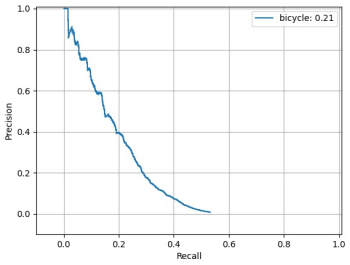
(d) Truck



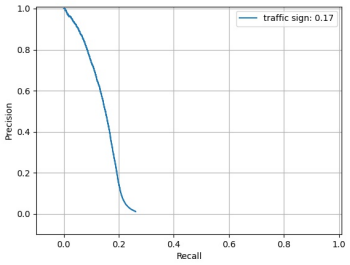
(e) Bus



(f) Motorcycle



(g) Bicycle



(h) Traffic Sign

Figure 6.10: Precision recall curves for each class in BDD100K dataset using SubEnsemble-SSD300 model with prior boxes tuned using BDD100K dataset

6.3.2 Quantification of Uncertainty in SSD Object Detector

To quantify the uncertainty in the detection made by SSD, we measure the probability score, entropy and bounding box deviation from all the 8732 detection boxes regressed by the object detection model.

Probability

Though we already used probability scores to evaluate OOD detection ability in Section **??** but the previous work done on uncertainty quantification methods [**?** **?** **?**] showed that the softmax scores extracted as explained in Section **??** proved to be effective in resulting a robust softmax scores for samples unknown to the trained model. Figure **??** shows the summary of the probability scores produced on both BDD100K and IDD dataset.

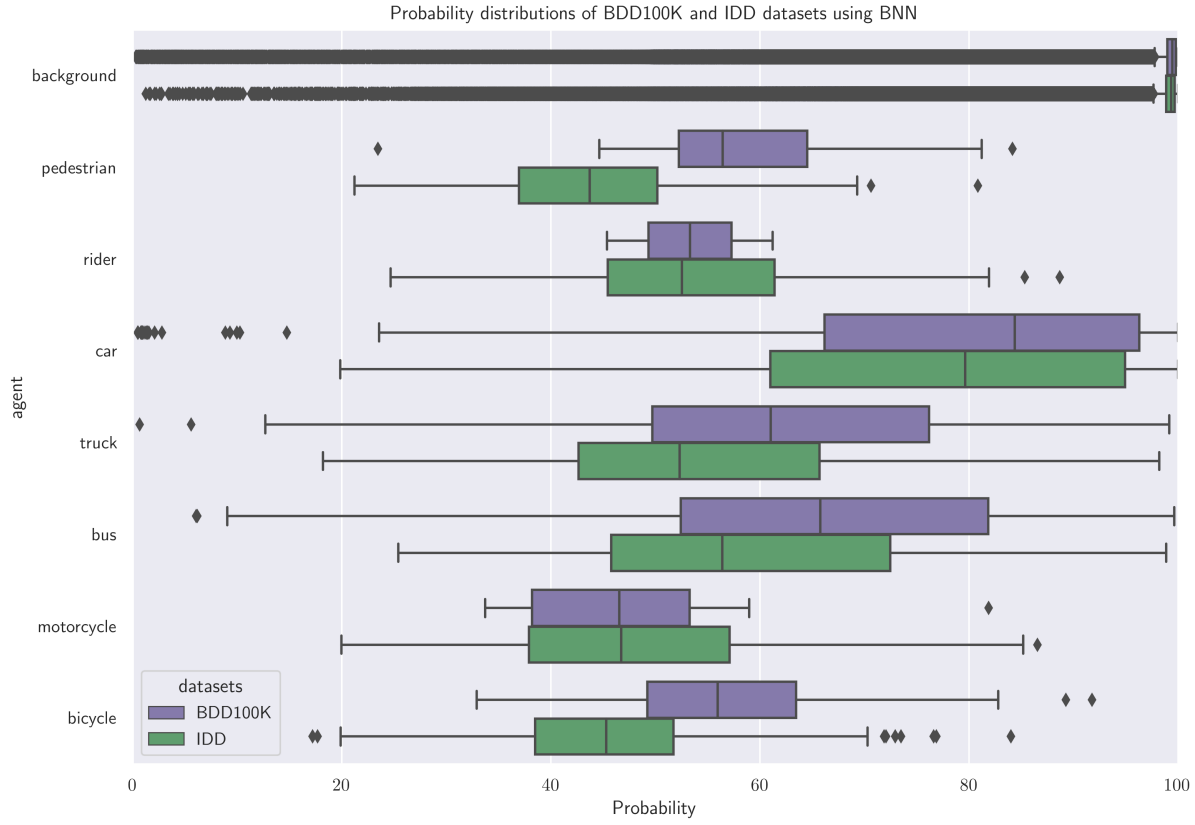


Figure 6.11: Box plot representing the summary of the probability calculated as the mean of the softmax scores from all the forward passes through the BNNs

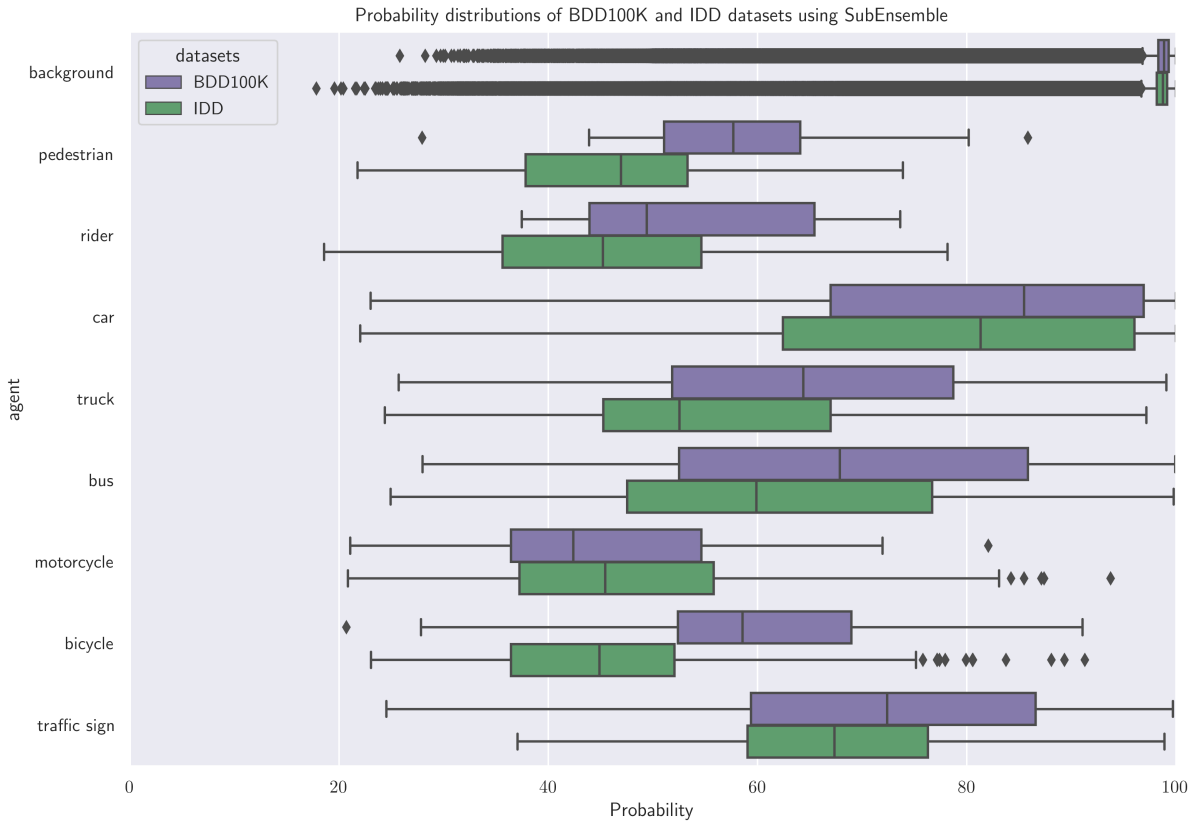


Figure 6.12: Box plot representing the summary of the probability calculated as the mean of the softmax scores from all the models in the Sub-Ensemble network

Our observations from the probability information shown in Figure **??** and Figure **??** are:

- The softmax scores for detections from BDD100K dataset are in general greater than the detections from IDD dataset.
- Softmax scores of background class are high because both BDD100K dataset and IDD dataset belong to the driving domain.
- It can be observed that the most similar object in BDD100K and IDD datasets like car and motorcycle has almost similar softmax score distributions.
- Since the number of traffic signs detected by BNNs in IDD dataset are ***Zero***, We decided not to consider the class for benchmarking purposes using BNNs.

Entropy

The entropy of detection is calculated from the mean softmax scores as mentioned in Section **??**. According to previous work proposed by **?**] used uncertainty in detecting OOD samples to solve classification tasks, the entropy is high in case of OOD samples.

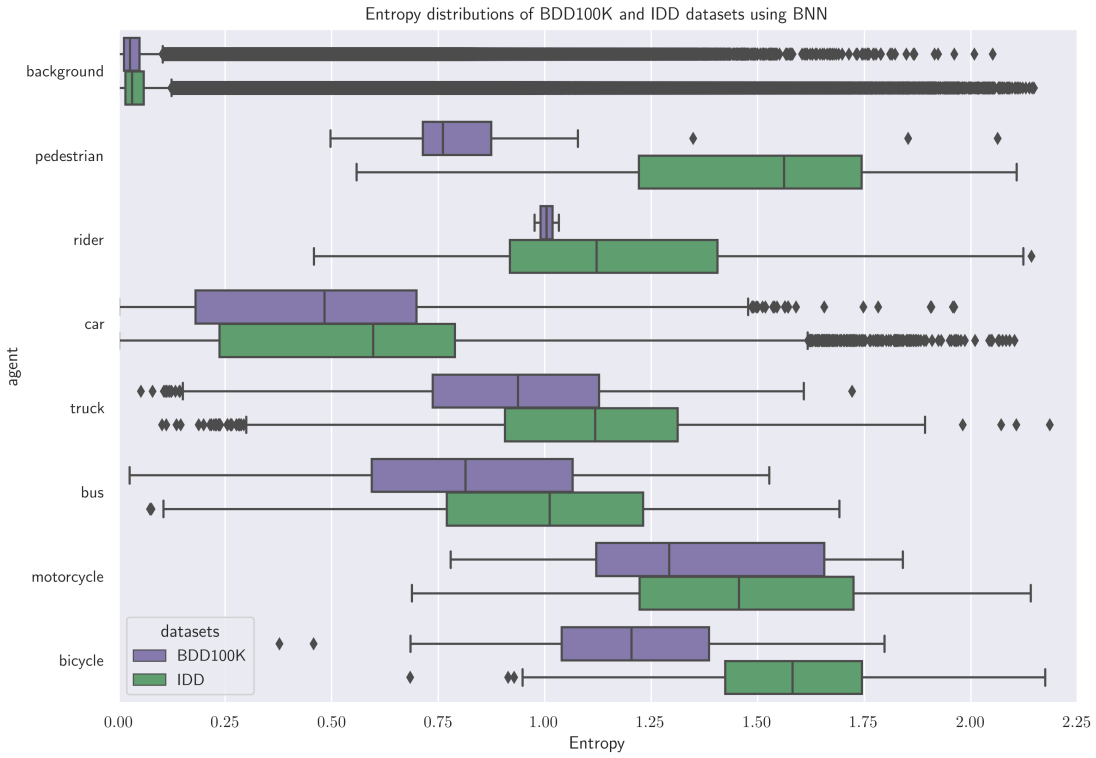


Figure 6.13: Box plot representing the summary of the entropy calculated from the mean of the softmax scores from all the forward passes through the BNNs

From the Figure **??** and Figure **??** which depicts the information about the distribution of entropies from the inferences made by the BNNs. The key observations are:

- Entropy differentiates the BDD100K and IDD data better than probability scores.
- The information presented also shows that entropy cannot differentiate the BDD100K and IDD datasets. This is due to the samples from IDD datasets are ambiguous compared to BDD100K dataset.
- Entropy is inherently not able to differentiate aleatoric uncertainty of ambiguous OOD samples and the epistemic uncertainty of pure OOD samples. This result is with the results from the work

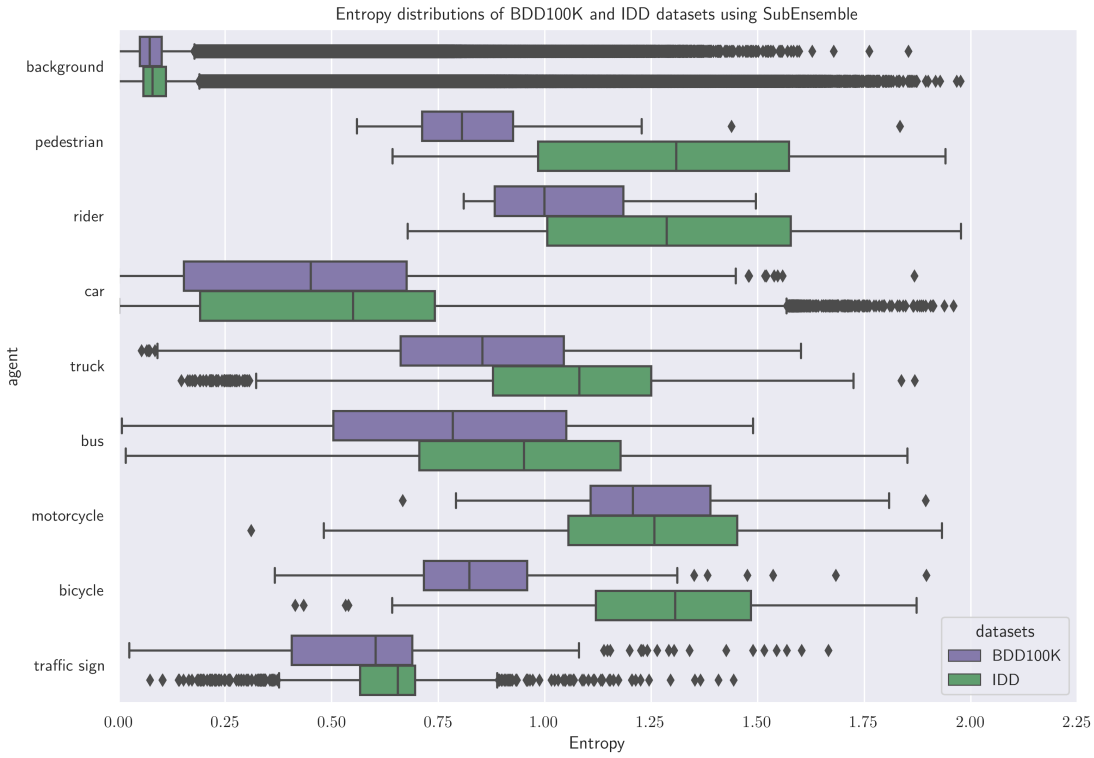


Figure 6.14: Box plot representing the summary of the entropy calculated from the mean of the softmax scores from all the models in the Sub-Ensemble networks

proposed by **?**]. The results show that entropy is not a good measure in detecting OOD samples which are dominated by ambiguous samples.

Box Deviation

Box Deviation is calculated as mentioned in Section **??**. This quantifies the inherent uncertainty in the regression head of the SSD object detection network. The bounding box deviations of both BDD100K and IDD datasets are shown in Figures **??** and **??**.

From the Figures **??** and **??**, we inferred that

- The distribution of box deviations is almost similar for both SubEnsembles and BNNs.
- BNNs produced higher box deviation values than Sub-Ensembles.
- The box deviations showed a higher ability of differentiation between BDD100K and IDD dataset.

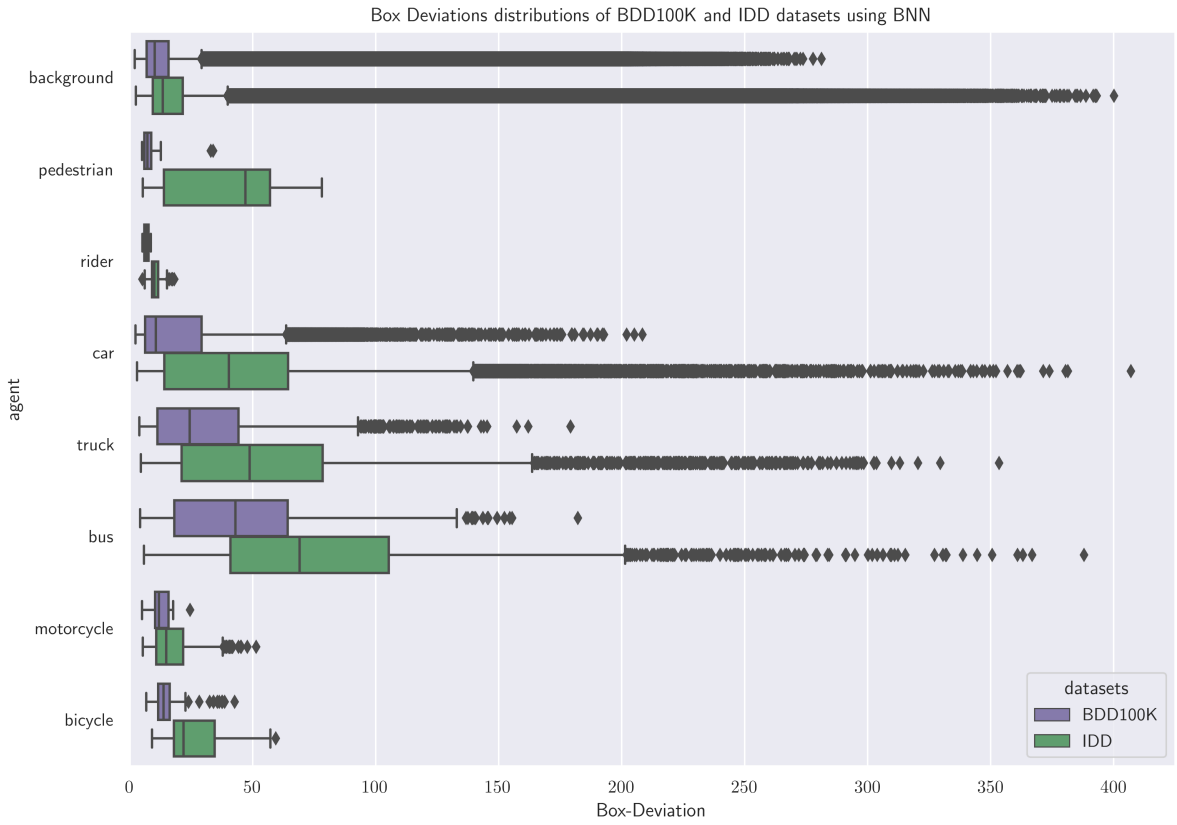


Figure 6.15: Box plot representing the summary of the box deviations calculated using the box detections from all the forward passes through the BNNs

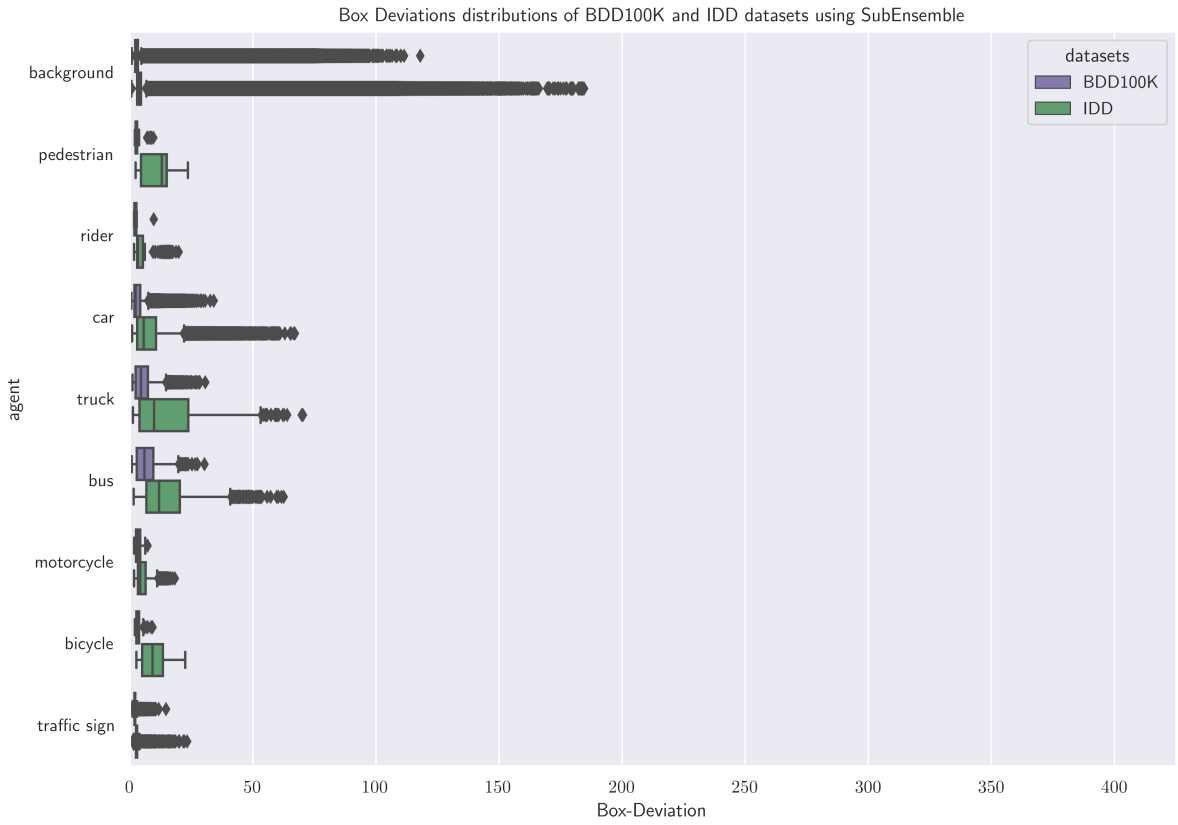
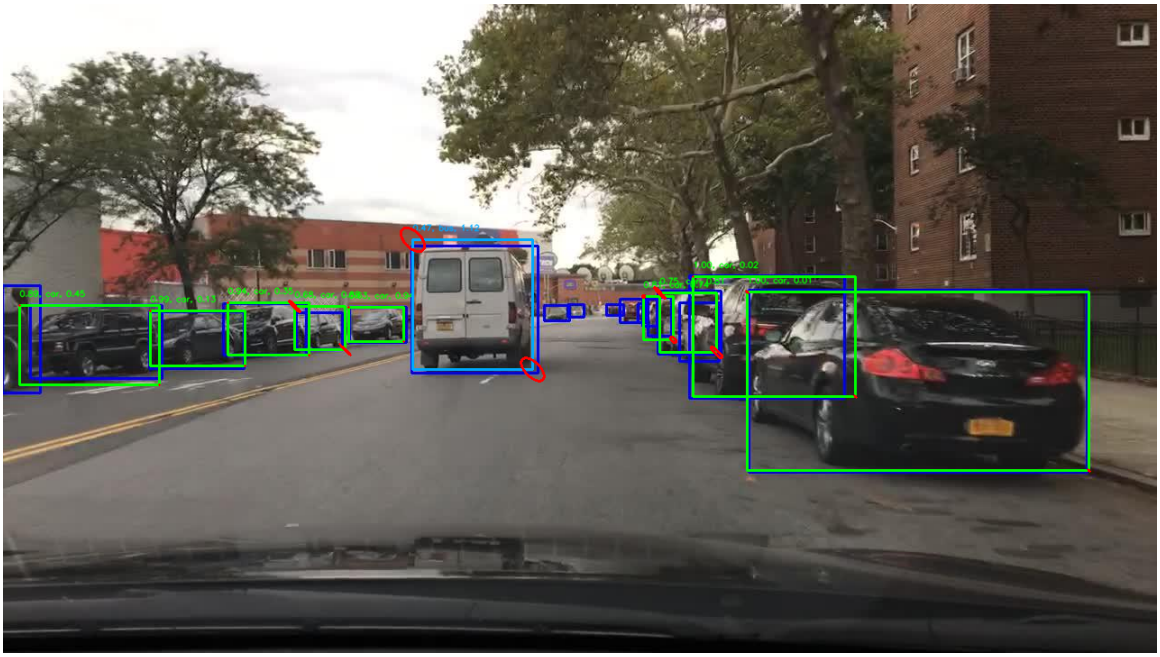
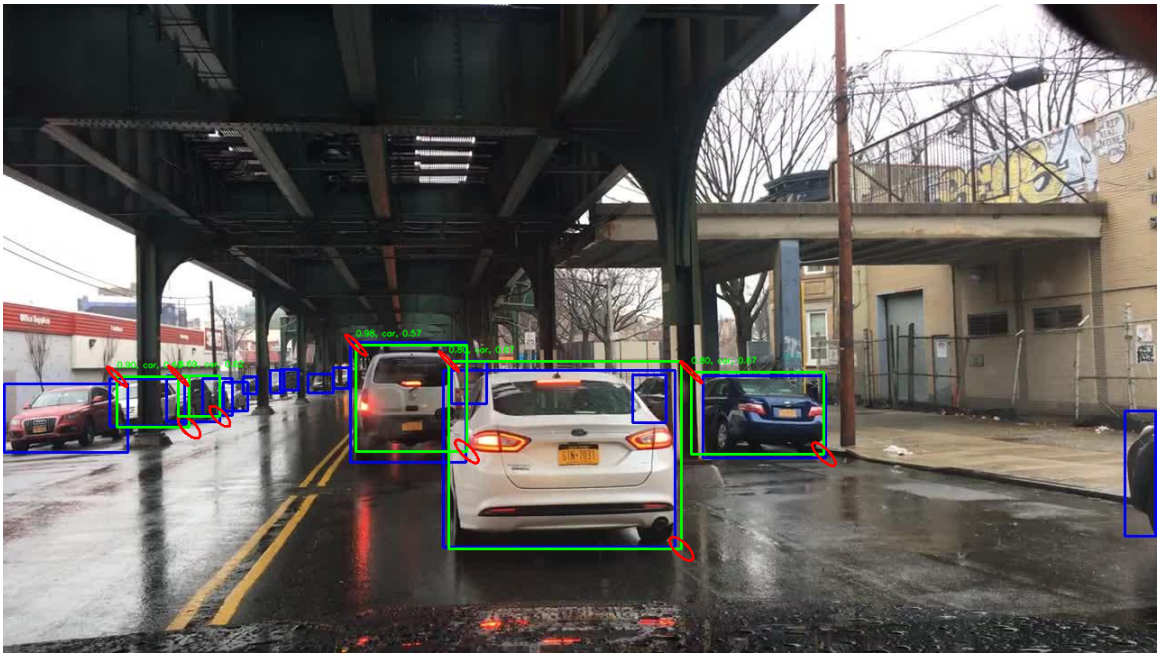


Figure 6.16: Box plot representing the summary of the entropy calculated from the mean of the softmax scores from multiple models of Sub-Ensemble.

The previous works done by **? ?**] showed that one cause for box deviation is the result of objects getting occluded, but from the samples, we observed from the IDD dataset that standard deviation is also high in case of objects not being an exact match to the objects experienced during the training procedure.

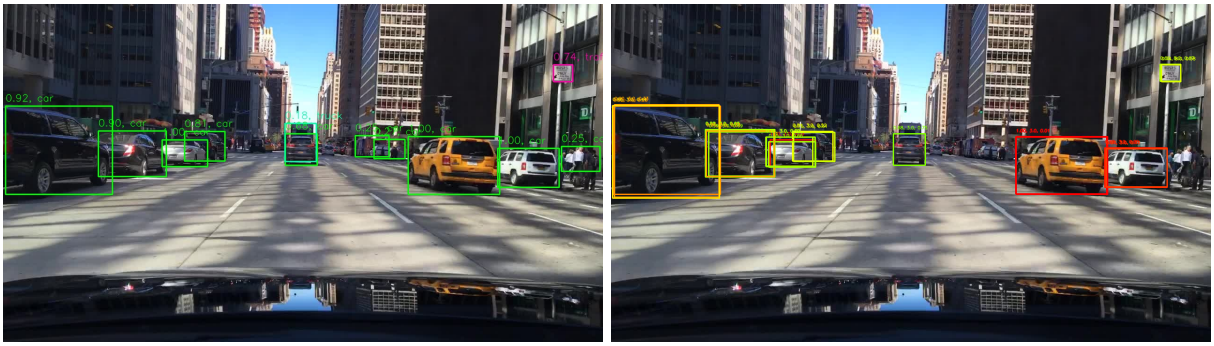


(a) Clear weather



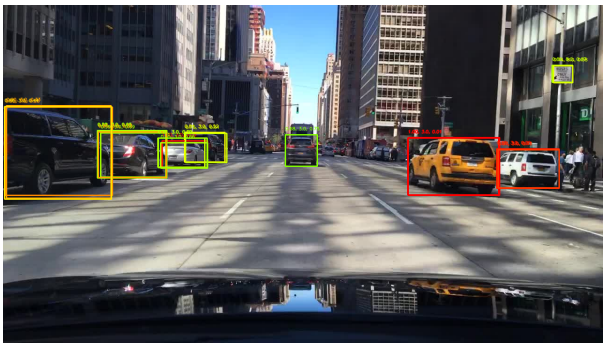
(b) Rainy weather

Figure 6.17: Detections made by SSD300 model on BDD100K dataset and the uncertainties quantified using Sub-Ensemble based SSD model. The boxes in blue are the ground truth boxes and the boxes in other colors are the detection made. The bounding box label of the detection is (probability, class name, entropy) and the major axis of the ellipse in red represents the box deviation. From Figure **??**, we can observe that detection of occluded cars are made with high variance. From Figure **??**, we can also observe that all detections are made with high variance.

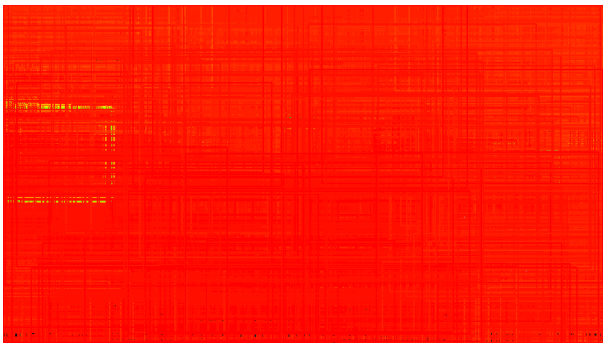


(a) Normal Detections

(b) Detections with entropy encoded in box color



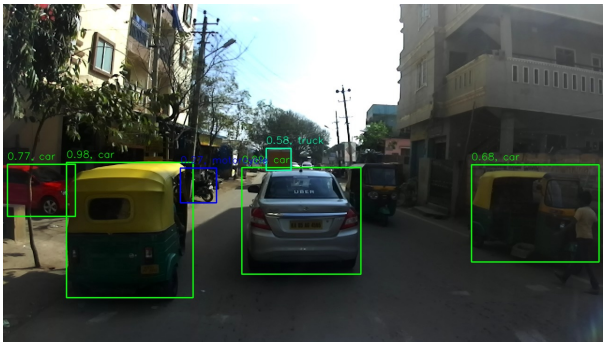
(c) Detections with variances represented as ellipses



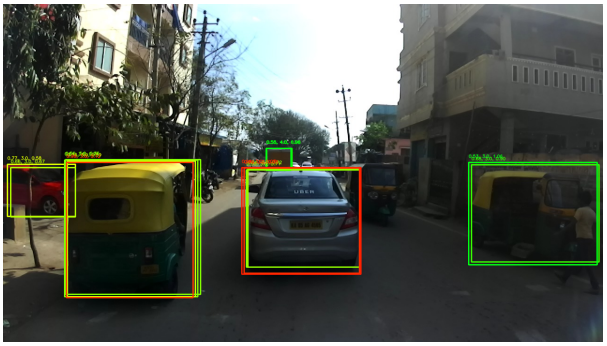
(d) Visualizing all 8732 boxes regressed by SSD300

(e) Colormap representing the entropy to box color encoding

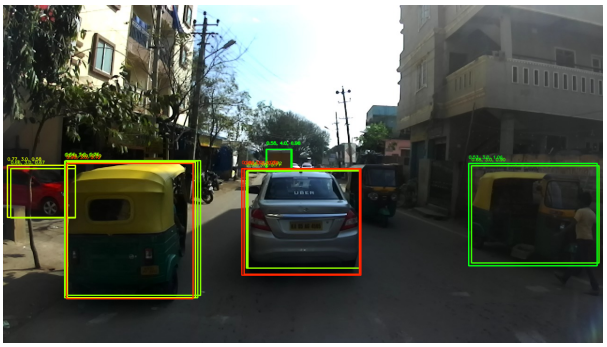
Figure 6.18: Inference of Bayesian-SSD300 model on a sample image from BDD100K, including visualization of various uncertainty quantification metrics chose to perform OOD detection. Since the image chosen is from ID dataset, there is very little confusion in boxes regressed from the images as shown in Figure **??**. An entropy value of 0 implies that the model is very sure about the class assigned to the bounding box and a maximum value suggests that the model is very unsure about the class of the object present. The bounding box label of the detection is (probability, class name, entropy) and the major axis of the ellipse in red represents the box deviation.



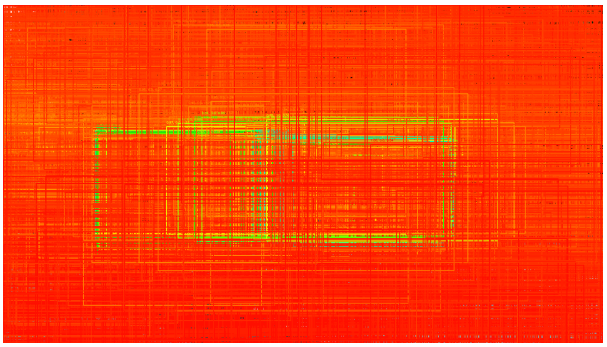
(a) Normal Detections



(b) Detections with entropy encoded in box color

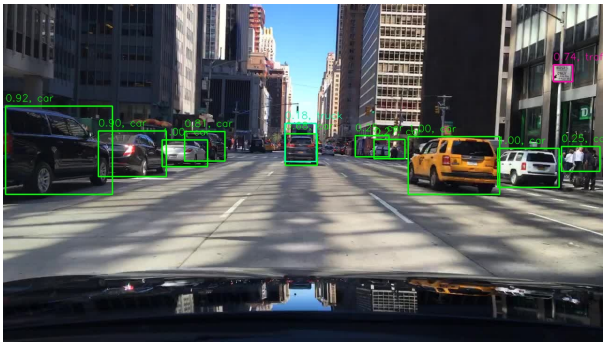


(c) Detections with variances represented as ellipses

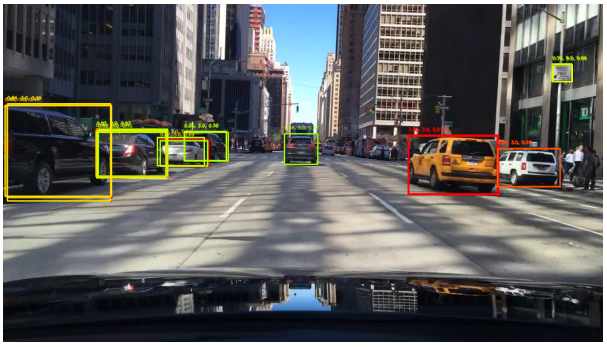


(d) Visualizing all 8732 boxes regressed by SSD300

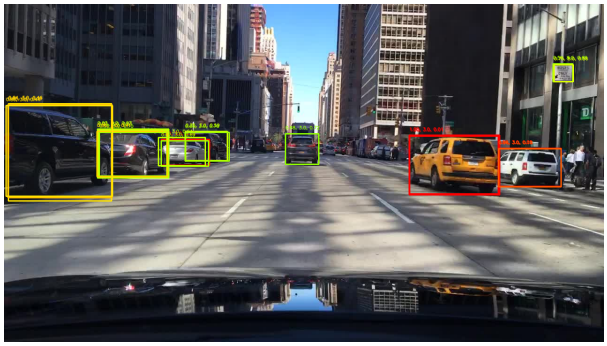
Figure 6.19: Inference of Bayesian-SSD300 model on a sample image from IDD and visualization of various uncertainty quantification metrics chose to perform OOD detection. Since the image chosen is from OOD dataset, there is a lot confusion in boxes regressed from the images as shown in Figure **??**.The bounding box label of the detection is (probability, class name, entropy) and the major axis of the ellipse in red represents the box deviation.



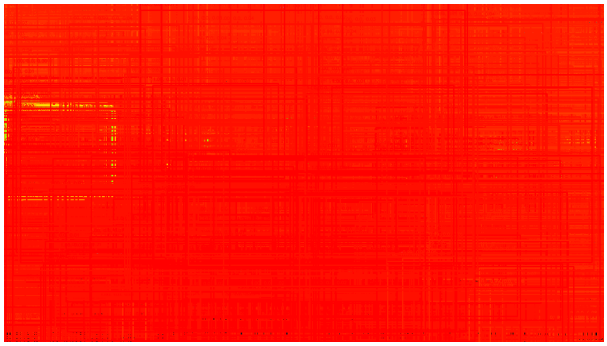
(a) Normal Detections



(b) Detections with entropy encoded in box color



(c) Detections with variances represented as ellipses



(d) Visualizing all 8732 boxes regressed by SSD300

Figure 6.20: Inference of Sub-Ensemble based SSD300 model and visualization of various uncertainty quantification metrics chose to perform OOD detection using . Since the image chosen is from ID dataset, there is very less confusion in boxes regressed from the images as shown in Figure **??**. The bounding box label of the detection is (probability, class name, entropy) and the major axis of the ellipse in red represents the box deviation.

From the experimental measurements and inference run visualization, it can be observed that:

- From Figure **??** we can infer that box deviation is high in case of occlusions, this behavior can be observed clearly from the top image. The deviations are high in the case of cars' partial visibility even though the distance from the ego vehicle is small.
- In case of OOD sample a Van which is a subclass of Car superclass is assigned high entropy and box deviation.
- We also observed a peculiar behavior of standard deviation being high, if there is a change in weather condition from the weather conditions in training data.
- BNNs even though generated a best mAP score has struggled in providing plausible uncertainty values both in ID and OOD cases. This sort of behavior is visualized in Figures **??** and **??**.
- Sub-Ensemble has performed well in quantifying uncertainty and also produced a good mAP score and only got outperformed by its Bayesian alternative.

(a) Normal Detections

(b) Detections with entropy encoded in box color

(c) Detections with variances represented as ellipses

(d) Visualizing all 8732 boxes regressed by SSD300

Figure 6.21: Inference of Sub-Ensemble based SSD300 model on an image sampled from OOD dataset and visualization of various uncertainty quantification metrics chose to perform OOD detection using Sub-Ensemble model. Since the image chosen is from OOD dataset, there is an inherent confusion in boxes regressed from the images as shown in Figure **??**. The bounding box label of the detection is (probability, class name, entropy) and the major axis of the ellipse in red represents the box deviation.

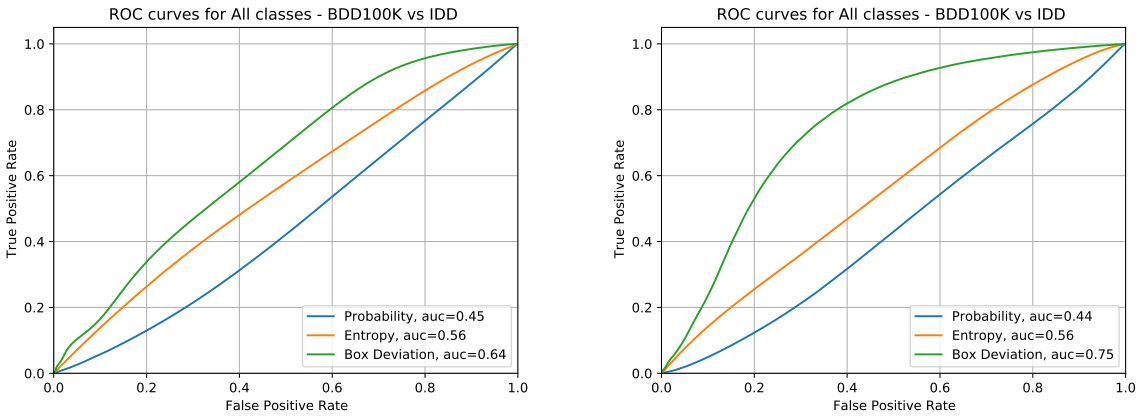
6.3.3 Out-of-Distribution Detection quantification using Uncertainty Quantification methods

In this experiment, we use the information about Probability, Entropy and Box-Deviation obtained in the previous experiment to quantify their ability to perform OOD classification. We use the method proposed in Section **??** to quantify the OOD classification ability. The metric we used to quantify the OOD ability is the Area Under the Receiver-Operating characteristic Curve.

Table 6.3: AUROC scores representing the metrics ability obtained from different uncertainty quantification models

Models			Bayesian - SSD300	Sub-Ensemble SSD300
Agents	Background	Probability	0.45	0.44
		Entropy	0.56	0.56
		Box-Deviation	**0.64**	**0.75**
	Pedestrian	Probability	0.14	0.21
		Entropy	0.88	0.86
		Box-Deviation	**0.93**	**0.95**
	Rider	Probability	0.75	0.34
		Entropy	0.59	0.72
		Box-Deviation	**0.82**	**0.84**
	Car	Probability	0.45	0.46
		Entropy	0.59	0.58
		Box-Deviation	**0.76**	**0.79**
	Truck	Probability	0.37	0.34
		Entropy	0.67	0.71
		Box-Deviation	**0.69**	**0.74**
	Bus	Probability	0.38	0.4
		Entropy	0.64	0.62
		Box-Deviation	**0.7**	**0.75**
	Motorcycle	Probability	0.49	0.53
		Entropy	0.6	0.51
		Box-Deviation	**0.62**	**0.71**
	Bicycle	Probability	0.23	0.19
		Entropy	0.84	0.88
		Box-Deviation	**0.85**	**0.94**
	Traffic Sign	Probability	**-**	0.41
		Entropy	**-**	0.62
		Box-Deviation	**-**	**0.88**
	All Classes	Probability	0.45	0.44
		Entropy	0.56	0.56
		Box-Deviation	**0.64**	**0.75**

From the graphs shown in Figure **??**, we can conclude that the Sub-Ensemble model with standard deviation metric out-performed all the other methods by a large margin. As already mentioned in Section **??**, the OOD dataset consists of various inter and intra-class variances in between them. Hence, it is very important to understand the performance of each uncertainty quantification model at each class level. To



(a) BNNs

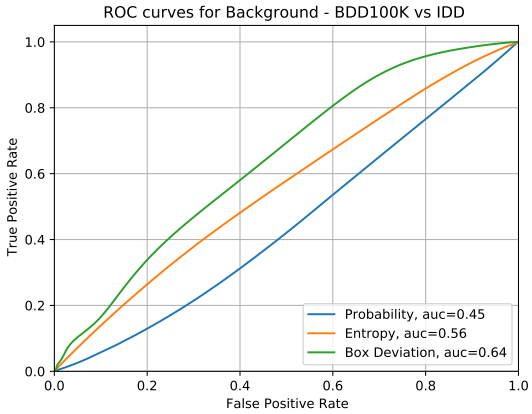
(b) Sub-Ensemble

Figure 6.22: Graphs visualizing the ROC curves representing the ability of OOD detection using BNNs and Sub-Ensemble based uncertainty quantification methods

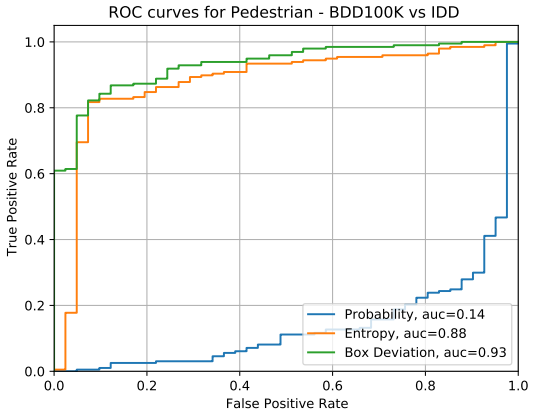
address this requirement, we used the per-class metrics to classify ID and OOD data and to quantification of this ability is shown in Table **??** and Figures **??** and **??**.

Observations

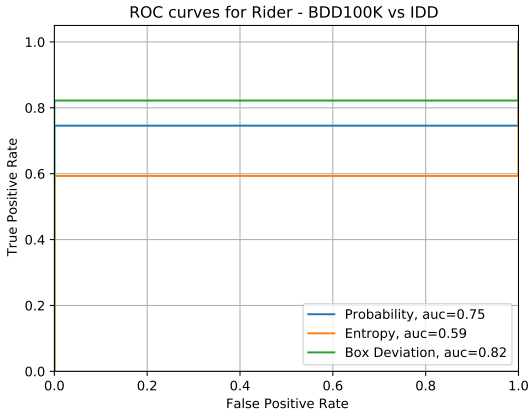
- All the metrics struggled in classifying background class between ID and OOD data. But Sub-Ensemble model detected OOD backgrounds representing that backgrounds does not match exactly because the data is collected in different countries.
- Entropy and Box deviation performed well in detecting OOD samples using both Bayesian and Sub-Ensemble model. Box deviation also performed well in this case.
- In case of Rider class, entropy struggled to detect because of the inherent ambiguity in between ID and OOD samples. But box deviation had performed well in classifying between BDD100K and IDD dataset.
- The performance of uncertainty quantification methods with Car, Truck and Bus classes is similar to that of Rider due to the inherent ambiguity between these classes in BDD100K and IDD datasets.
- Entropy and Box deviation had performed well in case of Bicycle class in both the methods.
- Overall, box deviation had outperformed entropy in classifying samples from BDD100K dataset to IDD dataset.



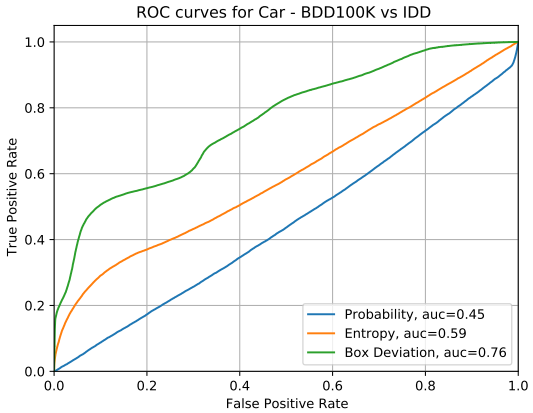
(a) Background



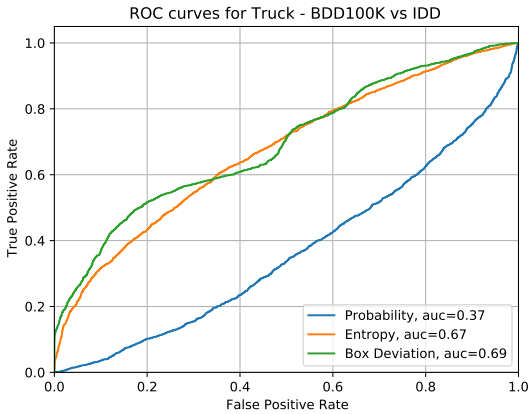
(b) Pedestrian



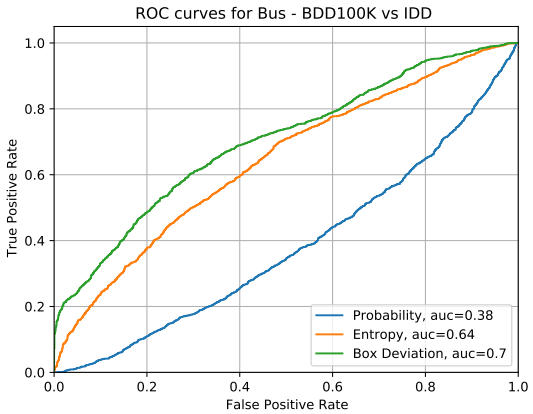
(c) Rider



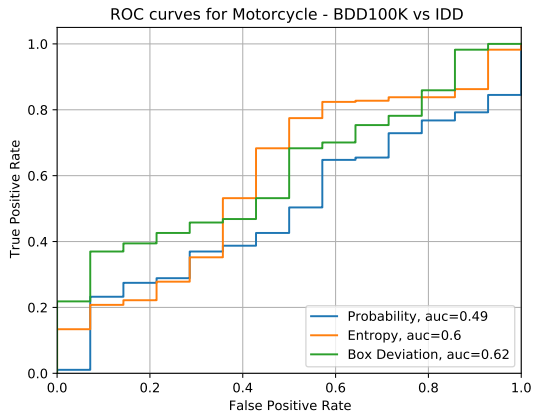
(d) Car



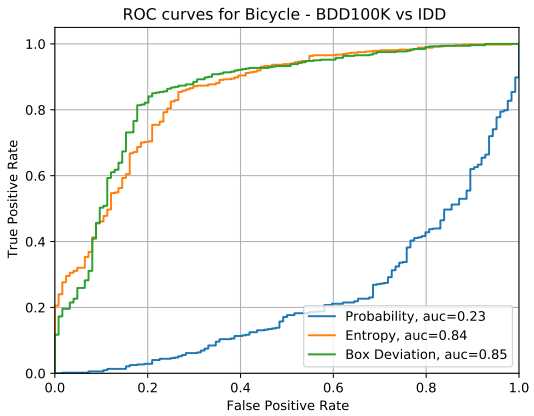
(e) Truck



(f) Bus

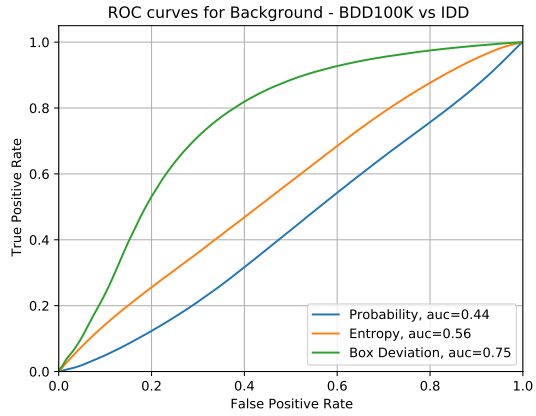


(g) Motorcycle

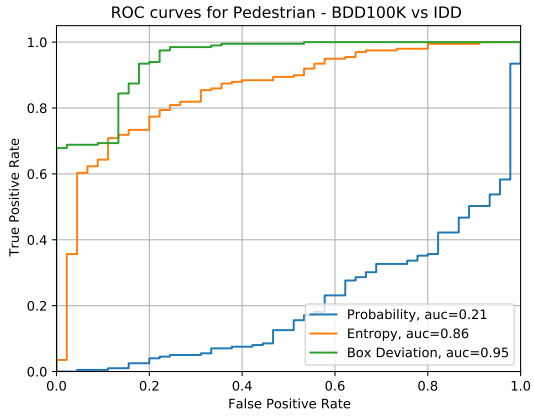


(h) Bicycle

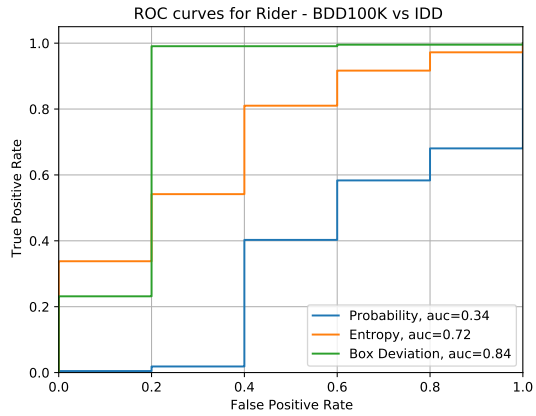
Figure 6.23: Receiver-Operating Curve curves for each class in BDD100K dataset using Bayesian-SSD300 model



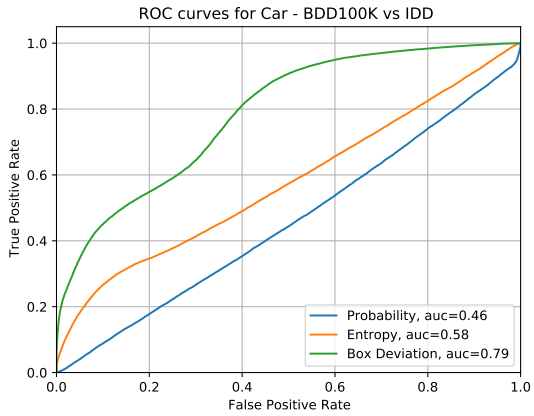
(a) Background



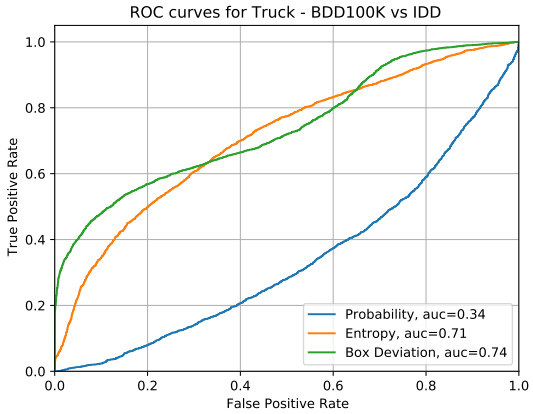
(b) Pedestrian



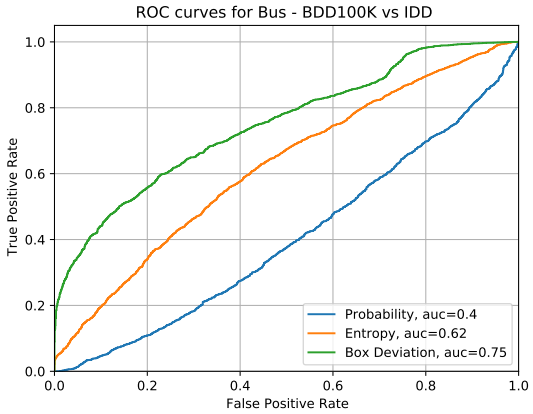
(c) Rider



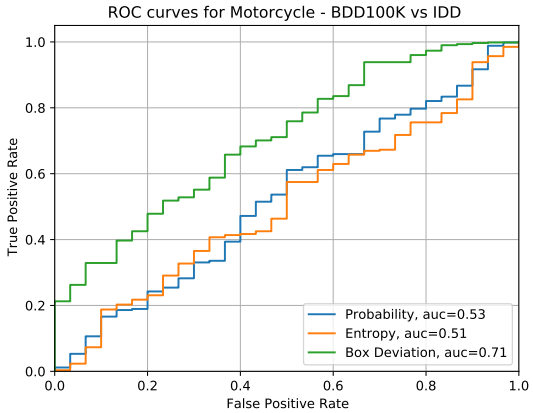
(d) Car



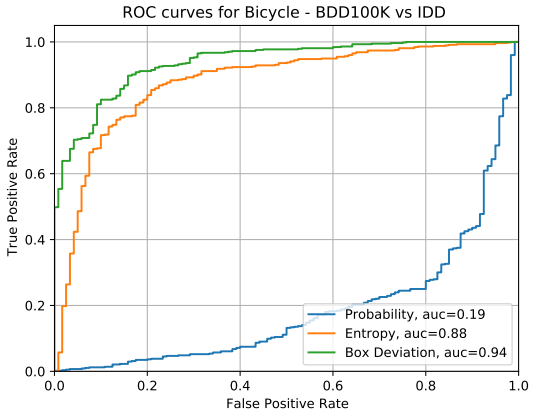
(e) Truck



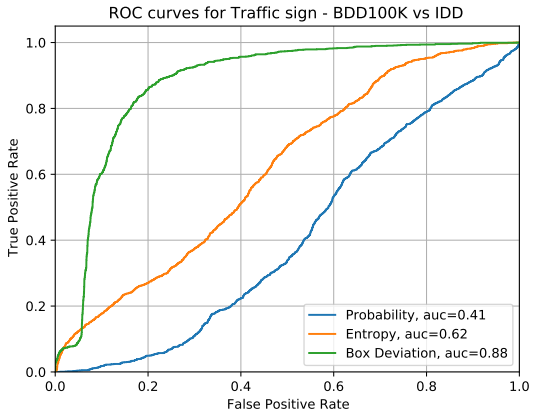
(f) Bus



(g) Motorcycle



(h) Bicycle



(i) Traffic Sign

Figure 6.24: Receiver-Operating Curve curves for each class in BDD100K dataset using SubEnsemble-SSD300 model

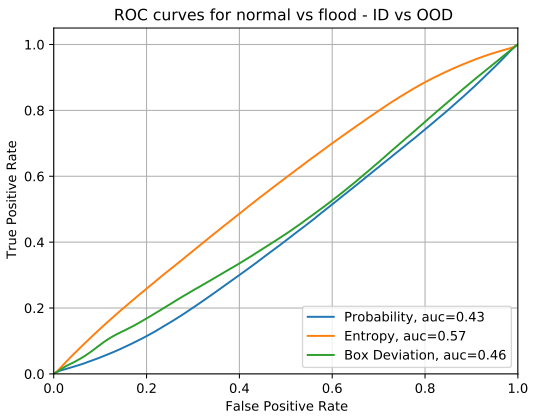
6.3.4 Out Of Distribution Detection with BDD100K-Weather dataset

This experiment is performed to quantify the ability of uncertainty quantification methods in classifying BDD100K dataset and BDD100K-weather dataset which is part of the OD^2 dataset proposed in Section **??**. We followed the same procedure as the previous experiments as outlined in Section **??**.

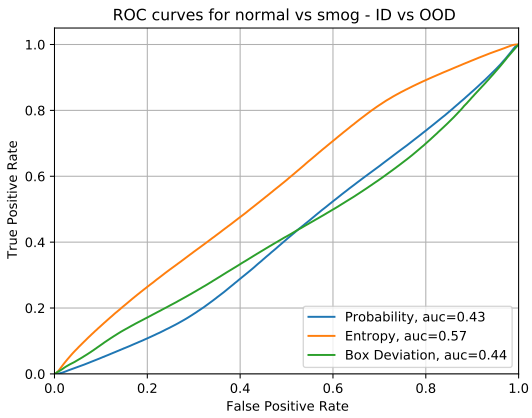
To understand the effectiveness of these methods with the different weather conditions, we evaluated the OOD detection ability in different weather conditions. The following graphs provide the quantification of the OOD detection ability

Table 6.4: Uncertainty quantification metrics calculated using Bayesian and Sub-Ensemble versions of SSD300 model. The scores suggest that uncertainty based OOD detectors are not effective in detecting OOD samples.

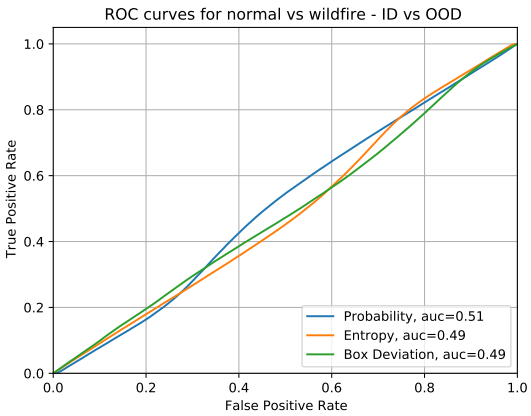
		Bayesian SSD300	Sub-Ensemble SSD300
Flood	Probability	0.43	0.5
	Entropy	**0.57**	0.5
	Box Deviation	0.46	0.5
Smog	Probability	0.43	0.5
	Entropy	**0.57**	0.5
	Box Deviation	0.44	0.5
Wild Fire	Probability	**0.51**	**0.53**
	Entropy	0.49	0.46
	Box Deviation	0.49	0.46



(a) Flood



(b) Smog



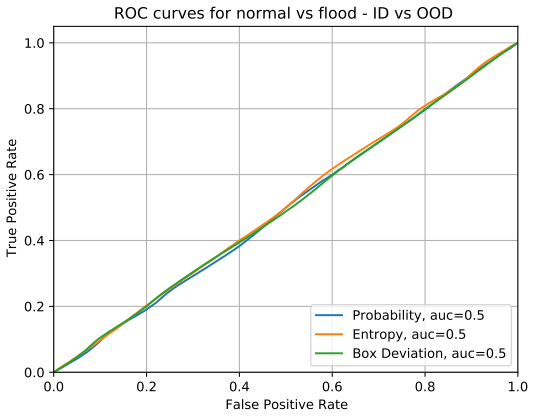
(c) Wild-Fire

Figure 6.25: Graphs visualizing the ROC curves representing the ability of OOD detection with BDD100K-Weather using BNNs

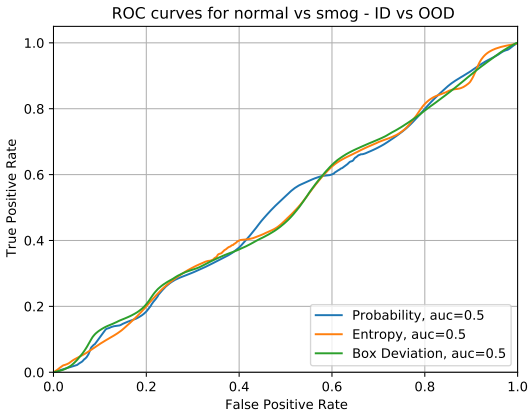
Figures **??** and **??** shows different images from different weathers placed column-wise with different metrics arranged row-wise. From these figures, we can observe that the weather condition neutralizes the variance as it makes all the samples ambiguous resulting in making entropy not useful for OOD detection.

Observations

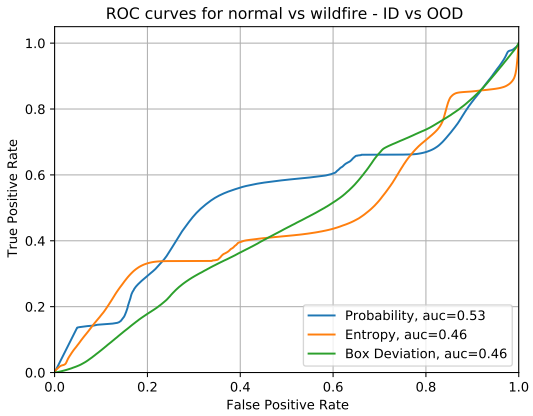
- The uncertainty quantification based OOD detection methods are not effective for detecting OOD data due to change in weather conditions.
- We observed that with change in weather the object detection performance has deteriorated especially in the case of flood images.
- The entropy visualizations suggest that the model is very confident on Smog dataset that un-modified dataset. This behavior needs to be further analyzed to extract a more plausible understanding of using entropy for OOD detection.
- The AUROC scores suggest that using uncertainty quantification methods performed almost similar to an unbiased random classifier.
- BNNs based uncertainty quantification could detect OOD data due to the object present in BDD100K-Weather dataset is not ambiguous compared to the training dataset.
- The reason for the uncertainty quantification methods not being able to detect skewed datasets complies with the results reported by **?**]. The authors performed their experiments on the classification dataset and showed that the entropy measured doesn't differentiate ID samples and their corresponding skewed samples.



(a) Flood



(b) Smog



(c) Wild-Fire

Figure 6.26: Graphs visualizing the ROC curves representing the ability of OOD detection with BDD100K-Weather using Sub-Ensembles

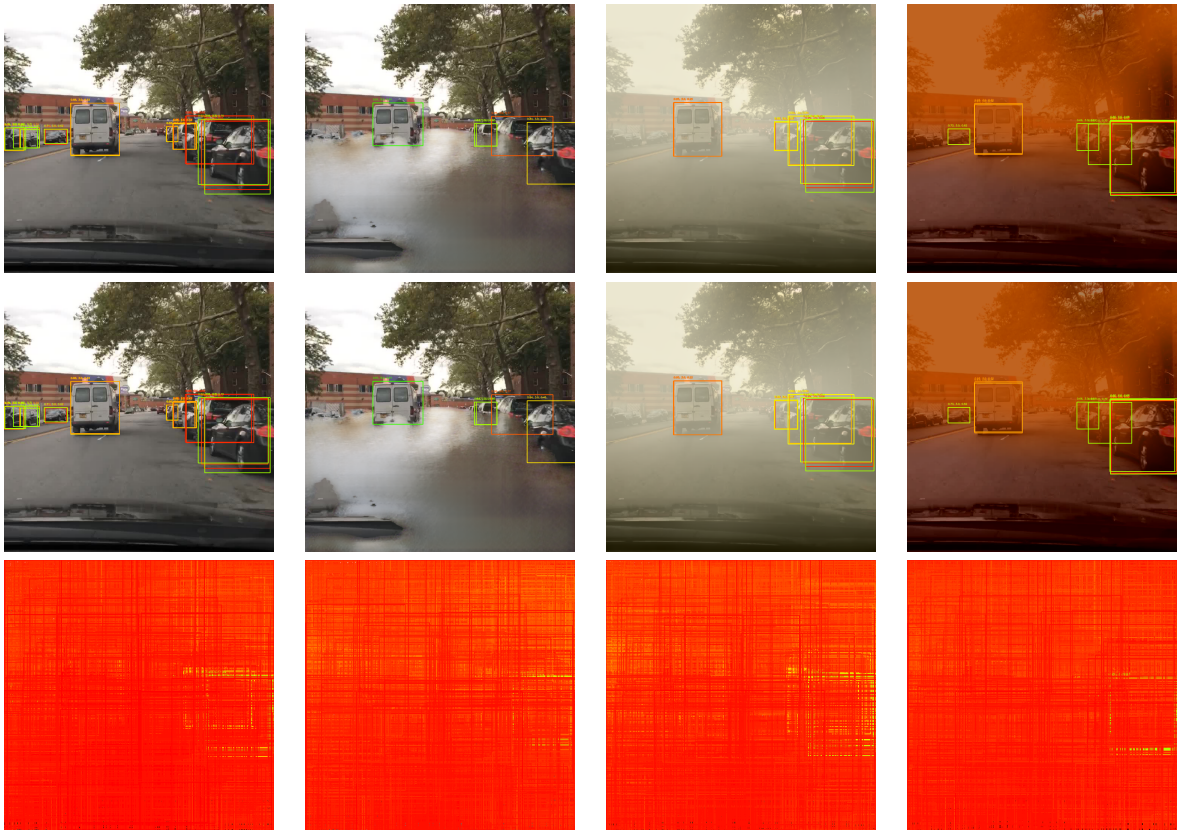


Figure 6.27: Detection and entropy visualization on a randomly sampled image from the BDD100K-Climate dataset using BNNs. We can observe that the model struggled in detecting objects as weather intensity increases. Also we can observe that there are lot less detection boxes seen compared to unchanged input image. The all boxes figure shows that majority of the boxes belong to background

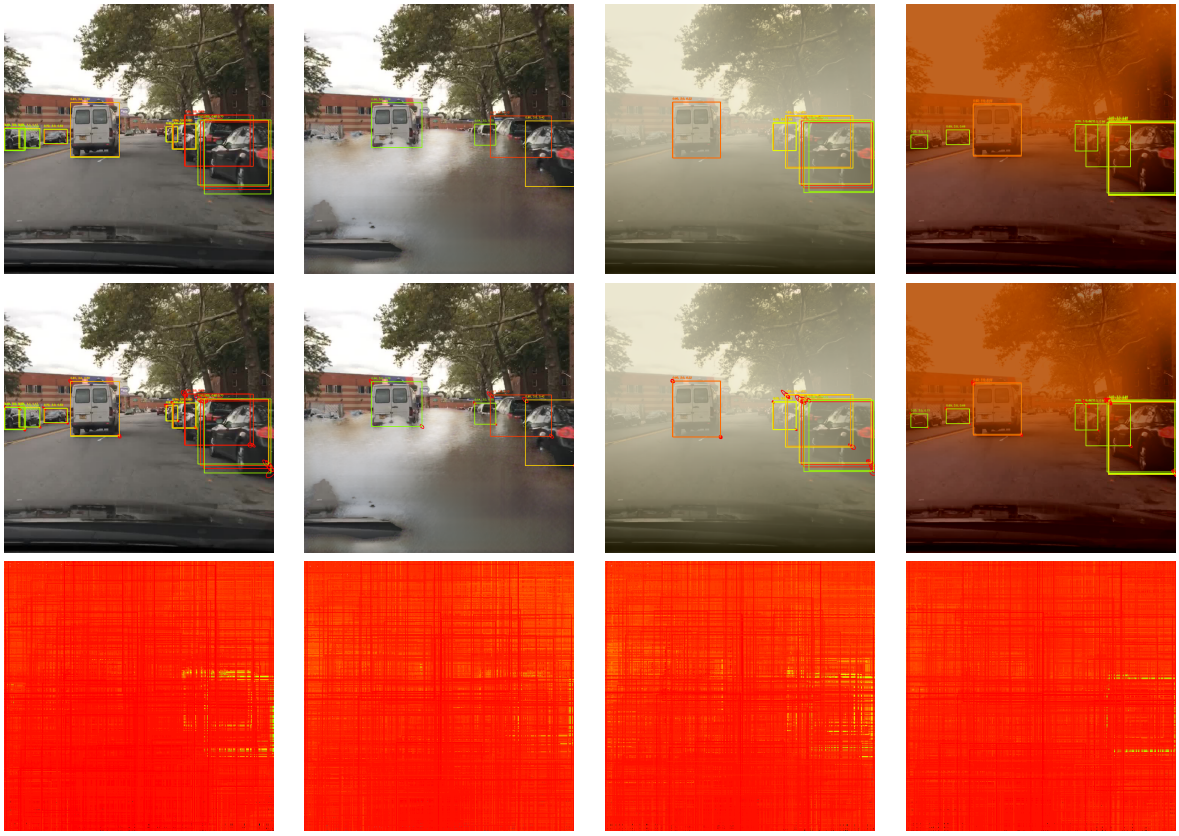


Figure 6.28: Detection and entropy visualization on a randomly sampled image from the BDD100K-Climate dataset using Sub-Ensembles. We observed a similar kind of box regression behavior to BNNs with sub-ensembles

7

Conclusions

Environment perception is a very important enabler for intelligent agents to operate autonomously without any human intervention. Autonomous Driving is one such application the is majorly being researched in both academia and industry. To ensure the safe operation of autonomous vehicles, the perception algorithms should be able to detect the samples that are not seen during training. The main focus of this work is to investigate the possibilities to perform Out Of Distribution (OOD) detection in the context of 2D Object Detection using image data.

7.1 Contributions

Our major contributions in this work are:

- A review of state-of-the-art methods for OOD detection on object classification and also discussed their abilities to adapt to the task of object detection is performed.
- We proposed a new benchmark dataset called Out-of-Distribution detection for Object Detection (OD^2) dataset to benchmark the OOD detection in 2D object detection to test the OOD abilities of proposed methods.
- Four different methods namely Max Softmax, ODIN, Mahalanobis distance based OOD detector, and uncertainty based OOD detector are chosen to perform OOD detection.
- To solve the task of object detection we trained a Single Shot multi-box Object Detector (SSD300) is trained on BDD100K dataset. To improve the performance of the SSD300 model on BDD100K dataset the scale and aspect ratios of the prior boxes are tuned.
- We explored the abilities of Max Softmax, ODIN, and Mahalanobis distance-based OOD detectors. From these three methods, we observed that ODIN outperformed the Max-Softmax based OOD detectors. Because of the large computational requirement to calculate covariance matrix we could not successfully model Mahalanobis distance-based OOD detector.
- To perform uncertainty quantification, we modelled a BNNs based SSD network and a Sub-Ensemble model of SSD object detector network. These Bayesian and Sub-Ensemble are modeled and trained.

- We used entropy to quantify uncertainty in the classification head of object detector and box deviation to quantify uncertainty in the regression head of object detectors.
- We performed extensive experimentation on all the three available OOD detectors for object detection purposes. We also performed studies on the class-specific behavior of the uncertainty quantification metrics.

We expect that the proposed OD^2 benchmark will further motivate other researchers to consider OOD performance during the development of object detection and other perception algorithms for autonomous driving, which we believe is an important component of future road safety.

7.2 Lessons Learned

From the extensive experimentation and evaluations we learnt that:

- The object detector performance is highly dependent on the prior knowledge tuned into the network based on the dataset.
- Deep learning-based object detectors struggle when deployed in open environments, where the occurrence of novel classes is highly plausible.
- The OOD detection methods proposed for classification did not directly transfer their performance into the task of object detection. The OOD methods heavily relied on the softmax layer in the classification head, this sort of reliance is not robust in the case of object detector networks.
- Uncertainty quantification methods proved to be robust in detecting such novel unseen objects proving the importance of their usage in safety-critical applications.
- Sub-Ensemble-based uncertainty quantification with box deviation as a metric had out-performed all other methods in OOD detection on the SSD model.
- After performing class-wise OOD detection studies, it can be inferred that entropy struggled in detecting samples that are ambiguous due to their semantic appearance.
- The OOD detection methods proposed in this work did not work as expected on the BDD100K-Climate data in the OD^2 benchmarking dataset proposed.

7.3 Future Work

The problem of OOD in the case of object detection is explored in this work. From the evaluations performed in this experiment, we suggest the following directions for further exploration:

- One area which plagued the concept of OOD in object detection is the presence of background class. The addition of this class automatically makes all the OOD samples to be classified as background due to the training strategy. An object detection method without the usage of background class would be a great contribution to the safe deployment of object detection models in safety-critical applications.

- Another area of improvement is exploring Two-stage networks for object detection purposes. The region proposal network used in these methods makes them effective for small object detection. So the OOD analysis can be effective.
- Exploring the effects of uncertainty calibration methods [**?**] on OOD detection is still a open-ended question.
- In this work, entropy and box-deviation are treated as two independent metrics. But combining both of these metrics to obtain a single novelty score might result in a better representation of OOD detection ability.
- Uncertainty calculated in this work is called predictive uncertainty and is not dis-entangled. This might have resulted in some fine information loss. As the semantic variation results in epistemic uncertainty and the quality of the image affects aleatoric uncertainty. Hence, we believe disentangling the uncertainty might result in better metrics for OOD detection.
- Though we believe a strong benchmark in the form of OD^2 dataset is proposed, we believe it can be further modified and extended to include more class-agnostic tasks.