Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Novelty detection for object detection in open set conditions

submitted by

Jim Martens

MIN Faculty

Department of Computer Science

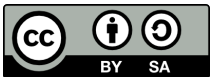
Course of studies: Computer Science M.Sc.

Matriculation Number: 6420323

Supervisor: Prof. Simone Frintrop

Co-Supervisor: Dr. Mikko Lauri

Licence



Acknowledgement

I would like to thank for the continued support, suggestions, and advice from my supervisor Prof. Dr. Simone Frintrop and co-supervisor Dr. Mikko Lauri.

Additionally, I would like to thank my friends and family for their continued support and helpful questions. Especially during some hard times their support was invaluable.

Furthermore, I am grateful for the Fridays for Future movement as it provides hope for a future worth fighting for. This hope was a light that allowed me to see the end of the tunnel, even if that felt light years away.

Lastly, I would like to thank society in general for providing me with the opportunity to study what I am interested in.

Abstract

Object detection in open set conditions is an important part of applying object detection to real world data sets. The key question is how to identify unknown or novel data. Dropout sampling with entropy thresholding is one possible avenue to answer the question.

This thesis compares vanilla SSD and SSD with dropout sampling on the MS COCO data set. The hyper parameters confidence threshold, entropy threshold, usage of non-maximum suppression, usage of dropout layers during testing, and the dropout keep ratio are varied to create different scenarios. Both results from macro and micro averaging are presented.

Results show a significant improvement of the object detection performance with a higher confidence threshold. The usage of non-maximum suppression improves performance as well. Usage of dropout layers during testing does not provide a better result, a lower keep ratio decreases the performance.

Contents

1 Introduction

The introduction first explains the wider context, before providing technical details.

Motivation

Famous examples like the automatic soap dispenser, which does not recognise the hand of a black person but dispenses soap when presented with a paper towel, raise the question of bias in computer systems [1]. Related to this ethical question regarding the design of so called algorithms is the question of algorithmic accountability [2].

Supervised neural networks learn from input-output relations and figure out by themselves what connections are necessary for that. This feature is also their Achilles heel: in effect, it makes them black boxes and prevents any answers to questions of causality.

However, these questions of causality are of enormous consequence when results of neural networks are used to make life changing decisions: is a correlation enough to bring forth negative consequences for a particular person? And if so, what is the possible defence against math? Similar questions can be raised when looking at computer vision networks that might be used together with so called smart CCTV cameras to discover suspicious activity.

This leads to the need for neural networks to explain their results. Such an explanation must come from the network or an attached piece of technology to allow mass adoption. Obviously, this setting poses the question of how such an endeavour can be achieved.

For neural networks there are fundamentally two types of problems: regression and classification. Regression deals with any case where the goal for the network is to come close to an ideal function that connects all data points. Classification, however, describes problems where the network is supposed to identify the class of any given input. In this thesis, I will work with both.

Object Detection in Open Set Conditions

More specifically, I will look at object detection in the open set conditions (see figure 1.1). In non-technical terms, this describes the conditions CCTV, and robots

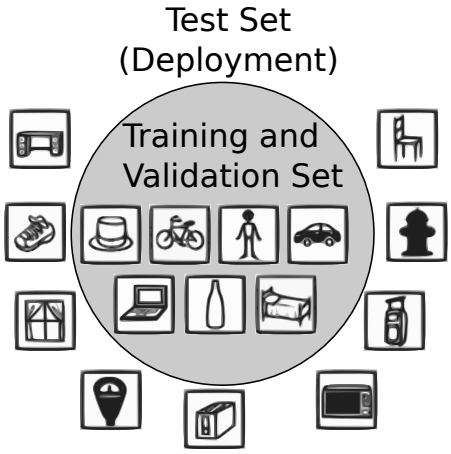


Figure 1.1: Open set problem: the test set contains classes that were not present during training time. Icons in this image have been taken from the COCO data set website (`https://cocodataset.org/#explore`) and were vectorised afterwards. Resembles figure 1 of Miller et al. [3].

outside of a laboratory operate in. In both cases images are recorded with cameras. In order to detect objects, a neural network has to analyse the images and return a list of detected and classified objects that it finds in the images. The problem here is that networks can only classify what they know. If presented with an object type that the network was not trained with, as happens frequently in real environments, it will still classify the object and might even have a high confidence in doing so. This is an example for a false positive. Anyone who uses the results of such a network could falsely assume that a high confidence means the classification is very likely correct. If one uses a proprietary system one might not even be able to find out that the network was never trained on a particular type of object. Therefore, it would be impossible for one to identify the output of the network as a false positive.

This reaffirms the need for automatic explanation. Such a system should recognise by itself that the given object is unknown and mark any classification result of the network as meaningless. Technically there are two slightly different approaches that deal with this type of task: model uncertainty and novelty detection.

Model uncertainty can be measured, for example, with dropout sampling. Dropout layers are usually used only during training, but Miller et al. [3] also use them during testing to achieve different results for the same image—making use of multiple forward passes. The output scores for the forward passes of the same image are then averaged. If the averaged class probabilities resemble a uniform distribution (every class has the same probability) this symbolises maximum uncertainty. Conversely, if there is one very high probability with every other being

very low, this signifies a low uncertainty. An unknown object is more likely to cause high uncertainty, which allows for an identification of false positive cases.

Novelty detection is another approach to solve the problem. In the realm of neural networks it is usually done with the help of auto-encoders that try to solve a regression problem of finding an identity function that reconstructs the given input [4]. Auto-encoders have, internally, at least two components: an encoder, and a decoder or generator. The job of the encoder is to find an encoding that compresses the input as well as possible, while simultaneously being as loss-free as possible. The decoder takes this latent representation of the input, and has to find a decompression that reconstructs the input as accurately as possible. During training these auto-encoders learn to reproduce a certain group of object classes. The actual novelty detection takes place during testing: given an image, and the output and loss of the auto-encoder, a novelty score is calculated. For some novelty detection approaches the reconstruction loss is the novelty score, others consider more factors. A low novelty score signals a known object. The opposite is true for a high novelty score.

Research Question

Auto-encoders work well for data sets like MNIST [5] but perform poorly on challenging real world data sets like MS COCO [6], complicating any potential comparison between them and object detection networks like Single Shot MultiBox Detector (SSD). Therefore, a comparison between model uncertainty with a network like SSD and novelty detection with auto-encoders is considered out of scope for this thesis.

Miller et al. [3] use an SSD pre-trained on COCO without further fine-tuning on the SceneNet RGB-D data set [7] and report good results regarding open set error (OSE) for an SSD variant with dropout sampling and entropy thresholding. If their results are generalisable it should be possible to replicate the relative difference between the variants on the COCO data set. This leads to the following hypothesis: *Dropout sampling delivers better object detection performance under open set conditions compared to object detection without it.*

For the purpose of this thesis, I use the vanilla SSD (as in: the original SSD) as baseline to compare against. In particular, vanilla SSD uses a per class confidence threshold of 0.01, an IOU threshold of 0.45 for the non-maximum suppression (NMS), and a top k value of 200. For this thesis, the top k value has been changed to 20 and the confidence threshold of 0.2 has been tried as well. The effect of an entropy threshold is measured against this vanilla SSD by applying entropy thresholds from 0.1 to 2.4 inclusive (limits taken from Miller et al.). Dropout sampling is compared to vanilla SSD with and without entropy thresholding.

Hypothesis Dropout sampling delivers better object detection performance under open set conditions compared to object detection without it.

Reader's Guide

First, chapter 2 presents related works, and provides the background for dropout sampling. Thereafter, chapter 3 explains how vanilla SSD works, how Bayesian SSD extends vanilla SSD, and how the decoding pipelines are structured. Chapter 4 presents the data sets, the experimental setup, and the results. This is followed by chapter 5, focusing on the discussion and closing.

The contribution of this thesis is found in chapters 3, 4, and 5.

2 Background

This chapter begins with an overview of previous works, followed by an explanation of the theoretical foundations of dropout sampling.

2.1 Related Works

The task of novelty detection can be accomplished in a variety of ways. Pimentel et al. [4] provide a review of novelty detection methods published over the previous decade. They showcase probabilistic, distance-based, reconstruction-based, domain-based, and information-theoretic novelty detection. Based on their categorisation, this thesis falls under reconstruction-based novelty detection as it deals only with neural network approaches. The other types of novelty detection will, therefore, only be introduced briefly.

2.1.1 Overview over types of Novelty Detection

Probabilistic approaches estimate the generative probabilistic density function (pdf) of the data. It is assumed that the training data is generated from an underlying probability distribution D. This distribution can be estimated with the training data, the estimate is defined as $\hat{D}$ and represents a model of normality. A novelty threshold is applied to $\hat{D}$ in a way that allows a probabilistic interpretation. Pidhorskyi et al. [8] combine a probabilistic approach to novelty detection with auto-encoders.

Distance-based novelty detection uses either nearest neighbour-based approaches (e.g. k-nearest neighbour [9]) or clustering-based approaches (e.g. k-means clustering algorithm [10]). Both methods are similar to estimating the pdf of data, they use well-defined distance metrics to compute the distance between two data points.

Domain-based novelty detection describes the boundary of the known data, rather than the data itself. Unknown data is identified by its position relative to the boundary. Support vector machines (e.g. implemented by Song et al. [11]) are a common implementation of this.

Information-theoretic novelty detection computes the information content of a data set, for example, with metrics like entropy. Such metrics assume that novel

data inside the data set significantly alters the information content of an otherwise normal data set. First, the metrics are calculated over the whole data set. Second, a subset is identified that causes the biggest difference in the metric when removed from the data set. This subset is considered to consist of novel data. For example, Filippone and Sanguinetti [12] provide a recent approach.

2.1.2 Reconstruction-based Novelty Detection

Reconstruction-based approaches use the reconstruction error in one form or another to calculate the novelty score. These can be auto-encoders that literally reconstruct the input but it also includes multilayer perceptron (MLP) networks which try to reconstruct the ground truth. Pimentel et al. [4] differentiate between neural network-based approaches and subspace methods. The first are further differentiated between MLPs, Hopfield networks, autoassociative networks, radial basis function, and self-organising networks. The remainder of this section focuses on MLP-based works, a particular focus will be on the task of object detection and Bayesian networks.

Novelty detection for object detection is intricately linked with open set conditions: the test data can contain unknown classes. Bishop [13] investigates the correlation between the degree of novel input data and the reliability of network outputs, and introduces a quantitative way to measure novelty.

The Bayesian approach provides a theoretical foundation for modelling uncertainty [14]. MacKay [15] provides a practical Bayesian framework for backpropagation networks. Neal [16] builds upon the work of MacKay and explores Bayesian learning for neural networks. However, these Bayesian neural networks do not scale well. Over the course of time, two major Bayesian approximations have been introduced: one based on dropout and one based on batch normalisation.

Gal and Ghahramani [17] show that dropout training is a Bayesian approximation of a Gaussian process. Subsequently, Gal [18] shows that dropout training actually corresponds to a general approximate Bayesian model. This means every network trained with dropout is an approximate Bayesian model. During inference the dropout remains active: this form of inference is called Monte Carlo Dropout (MCDO). Miller et al. [3] build upon the work of Gal and Ghahramani: they use MCDO under open-set conditions for object detection. In a second paper [19], Miller et al. continue their work and compare merging strategies for sampling-based uncertainty techniques in object detection.

Teye et al. [20] make the point that most modern networks have adopted other regularisation techniques. Ioffe and Szeged [21] introduce batch normalisation which has been adapted widely in the meantime. Teye et al. show how batch normalisation training is similar to dropout and can be viewed as an approximate Bayesian inference. Estimates of the model uncertainty can be gained with a

technique named Monte Carlo Batch Normalisation (MCBN). Consequently, this technique can be applied to any network that utilises standard batch normalisation. Li et al. [22] investigate the problem of poor performance when combining dropout and batch normalisation: dropout shifts the variance of a neural unit when switching from train to test; batch normalisation does not change the variance. This inconsistency leads to a variance shift which can have a larger or smaller impact based on the network used.

Non-Bayesian approaches have also been developed. Usually they are compared with MCDO and show better performance. Postels et al. [23] provide a sampling-free approach for uncertainty estimation that does not affect training, and approximates the sampling at test time. They compare it to MCDO and find less computational overhead with better results. Lakshminarayanan et al. [24] implement a predictive uncertainty estimation using deep ensembles. Compared to MCDO, it shows better results. Geifman et al. [25] introduce an uncertainty estimation algorithm for non-Bayesian deep neural classification that estimates the uncertainty of highly confident points using earlier snapshots of the trained model and improves, among others, the approach introduced by Lakshminarayanan et al. Sensoy et al. [26] explicitly model prediction uncertainty: a Dirichlet distribution is placed over the class probabilities. Consequently, the predictions of a neural network are treated as subjective opinions.

In addition to the aforementioned Bayesian and non-Bayesian works, there are some Bayesian works that do not quite fit with the rest but are important as well. Mukhoti and Gal [27] contribute metrics to measure uncertainty for semantic segmentation. Wu et al. [28] introduce two innovations that turn variational Bayes into a robust tool for Bayesian networks: first, a novel deterministic method to approximate moments in neural networks which eliminates gradient variance, and second, a hierarchical prior for parameters and an empirical Bayes procedure to select prior variances.

2.2 Background for Dropout Sampling

This section will use the **notation** defined in table 2.1 on page 8. The subsections 'Dropout Variational Inference' and 'Dropout Sampling for Object Detection' follow the theoretical explanation by Miller et al. [3]. To understand dropout sampling, it is necessary to explain the idea of Bayesian neural networks. They place a prior distribution over the network weights, for example a Gaussian prior distribution: $\mathbf{W} \sim \mathcal{N}(0, I)$. In this example $\mathbf{W}$ are the weights and I symbolises that every weight is drawn from an independent and identical distribution. The training of the network determines a plausible set of weights by evaluating the probability output (posterior) over the weights given the training data $\mathbf{T}$:

Table 2.1: Notation for background

symbol	meaning
$\mathbf{W}$	weights
$\mathbf{T}$	training data
$\mathcal{N}(0, I)$	Gaussian distribution
I	independent and identical distribution
$p(\mathbf{W}\|\mathbf{T})$	probability of weights given training data
$\mathcal{I}$	an image
$\mathbf{q} = p(y\|\mathcal{I}, \mathbf{T})$	probability of all classes given image and training data
$H(\mathbf{q})$	entropy over probability vector
$\widetilde{\mathbf{W}}$	weights sampled from $p(\mathbf{W}\|\mathbf{T})$
$\mathbf{b}$	bounding box coordinates
$\mathbf{s}$	softmax scores
$\overline{\mathbf{s}}$	averaged softmax scores
D	detections of one forward pass
$\mathfrak{D}$	set of all detections over multiple forward passes
$\mathcal{O}$	observation
$\overline{\mathbf{q}}$	probability vector for observation

$p(\mathbf{W}|\mathbf{T})$. However, this evaluation cannot be performed in any reasonable amount of time. Therefore approximation techniques are required. In those techniques the posterior is fitted with a simple distribution $q_\theta^*(\mathbf{W})$. The original and intractable problem of averaging over all weights in the network is replaced with an optimisation task: the parameters of the simple distribution are optimised [29].

Dropout Variational Inference

Kendall and Gal [29] show an approximation for classfication and recognition tasks. Dropout variational inference is a practical approximation technique by adding dropout layers in front of every weight layer and also using them during test time to sample from the approximate posterior. In effect, this results in the approximation of the class probability $p(y|\mathcal{I}, \mathbf{T})$ by performing n forward passes through the network and averaging the so obtained softmax scores $\mathbf{s}_i$, given an image $\mathcal{I}$ and the training data $\mathbf{T}$:

$$p(y|\mathcal{I}, \mathbf{T}) = \int p(y|\mathcal{I}, \mathbf{W}) \cdot p(\mathbf{W}|\mathbf{T}) d\mathbf{W} \approx \frac{1}{n} \sum_{i=1}^{n} \mathbf{s}_i \tag{2.1}$$

With this dropout sampling technique, n model weights $\widetilde{\mathbf{W}}_i$ are sampled from the posterior $p(\mathbf{W}|\mathbf{T})$. The class probability $p(y|\mathcal{I}, \mathbf{T})$ is a probability vector $\mathbf{q}$

over all class labels. Finally, the uncertainty of the network with respect to the classification is given by the entropy $H(\mathbf{q}) = -\sum_i q_i \cdot \log q_i$.

Dropout Sampling for Object Detection

Miller et al. [3] apply the dropout sampling to object detection. In that case $\mathbf{W}$ represents the learned weights of a detection network, for example SSD [30]. Every forward pass uses a different network $\widetilde{\mathbf{W}}$ which is approximately sampled from $p(\mathbf{W}|\mathbf{T})$. Each forward pass in object detection results in a set of detections, each consisting of bounding box coordinates $\mathbf{b}$ and softmax score $\mathbf{s}$. The detections are denoted by Miller et al. as $D_i = \{\mathbf{s}_i, \mathbf{b}_i\}$. The detections of all passes are put into a large set $\mathfrak{D} = \{D_1, ..., D_2\}$.

All detections with mutual intersection-over-union scores (IoU) of 0.95 or higher are defined as an observation $\mathcal{O}_i$. Subsequently, the corresponding vector of class probabilities $\overline{\mathbf{q}}_i$ for the observation is calculated by averaging all score vectors $\mathbf{s}_j$ in a particular observation $\mathcal{O}_i$: $\overline{\mathbf{q}}_i \approx \overline{\mathbf{s}}_i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{s}_j$. The label uncertainty of the detector for a particular observation is measured by the entropy $H(\overline{\mathbf{q}}_i)$.

If $\overline{\mathbf{q}}_i$ resembles a uniform distribution the entropy will be high. A uniform distribution means that no class is more likely than another, which is a perfect example of maximum uncertainty. Conversely, if one class has a very high probability the entropy will be low.

In open set conditions it can be expected that falsely generated detections for unknown object classes have a higher label uncertainty. A threshold on the entropy $H(\overline{\mathbf{q}}_i)$ can then be used to identify and reject these false positive cases.

3 Methods

This chapter explains the functionality of vanilla SSD, Bayesian SSD, and the decoding pipelines.

3.1 Vanilla SSD

Vanilla SSD is based upon the VGG-16 network (see figure 3.1) and adds extra feature layers. The entire image (always size 300x300) is divided up into so called anchor boxes. During training, each of these boxes is mapped to a ground truth box or background. For every anchor box both the offset to the object and the class confidences are calculated. The output of the SSD network are the predictions with class confidences, offsets to the anchor box, anchor box coordinates, and variance. The model loss is a weighted sum of localisation and confidence loss. As the network has a fixed number of anchor boxes, every forward pass creates the same number of detections—8732 in the case of SSD 300x300.

Notably, the object proposals are made in a single run for an image—single shot. Other techniques like Faster R-CNN employ region proposals and pooling. For more detailed information on SSD, please refer to Liu et al. [30].

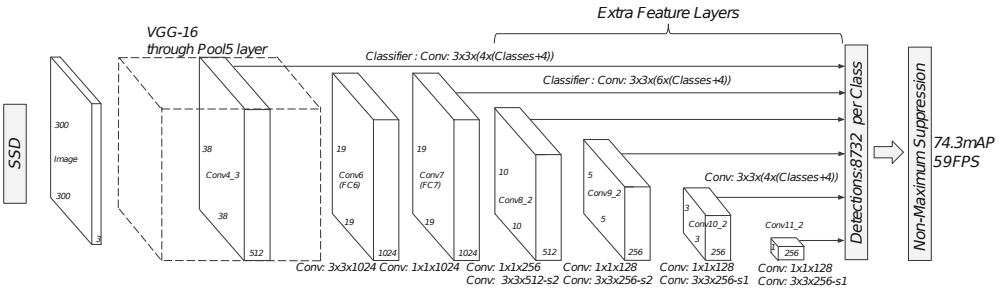


Figure 3.1: The vanilla SSD network as defined by Liu et al. [30]. VGG-16 is the base network, extended with extra feature layers. These predict offsets to anchor boxes with different sizes and aspect ratios. Furthermore, they predict the corresponding confidences.

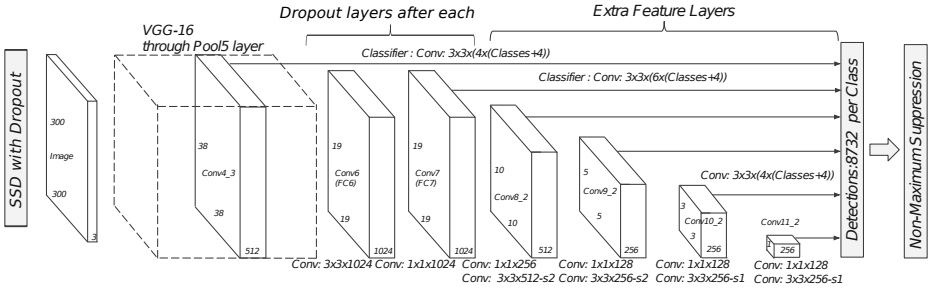


Figure 3.2: The Bayesian SSD network as defined by Miller et al. [3]. It adds dropout layers after the fc6 and fc7 layers.

3.2 Bayesian SSD for Model Uncertainty

Networks trained with dropout are a general approximate Bayesian model [18] and can be used for everything a true Bayesian model could be used for. Miller et al. apply this idea to SSD: two dropout layers are added to vanilla SSD, after the layers fc6 and fc7 respectively (see figure 3.2).

Motivation for this is model uncertainty: for the same object on the same image, an uncertain model will predict different classes across multiple forward passes. This uncertainty is measured with entropy: every forward pass results in predictions, these are partitioned into observations, and subsequently their entropy is calculated. A higher entropy indicates a more uniform distribution of confidences whereas a lower entropy indicates a larger confidence in one class and very low confidences in other classes.

3.2.1 Implementation Details

For this thesis, an SSD implementation based on Tensorflow [31] and Keras [32] is used. It has been modified to support entropy thresholding, partitioning of observations, and dropout layers in the SSD model. Entropy thresholding takes place before the per class confidence threshold is applied.

The Bayesian variant was not fine-tuned and operates with the same weights that vanilla SSD uses as well.

3.3 Decoding Pipelines

The raw output of SSD is not very useful: it contains thousands of boxes per image. Among them are many boxes with very low confidences or background classifications, those need to be filtered out to get any meaningful output of the

network. The process of filtering is called decoding and presented for the three structural variants of SSD used in the thesis.

3.3.1 Vanilla SSD

Liu et al. [30] use Caffe for their original SSD implementation. The decoding process contains largely two phases: decoding and filtering. Decoding transforms the relative coordinates predicted by SSD into absolute coordinates. Before decoding, the shape of the output per batch is $(batch\_size, \#nr\_boxes, \#nr\_classes + 12)$. The last twelve elements are split into the four bounding box offsets, the four anchor box coordinates, and the four variances; there are 8732 boxes. After decoding, of the twelve elements only four remain: the absolute coordinates of the bounding box.

Filtering of these boxes is first done per class: all classes except background are iterated through and for every detection the class id of the current class, the confidence for that class, and the bounding box coordinates are kept. The filtering consists of confidence thresholding and a subsequent non-maximum suppression (NMS). All boxes that pass NMS are added to a per image maxima list. One box could make the confidence threshold for multiple classes and, hence, be present multiple times in the maxima list for the image. In the end, a total of k boxes with the highest confidences is kept per image across all classes. The original implementation uses a confidence threshold of 0.01, an IOU threshold for NMS of 0.45 and a top k value of 200.

The vanilla SSD per class confidence threshold and NMS has one weakness: even if SSD correctly predicts all objects as the background class with high confidence, the per class confidence threshold of 0.01 will consider predictions with very low confidences; as background boxes are not present in the maxima collection, many low confidence boxes can be. Furthermore, the same detection can be present in the maxima collection for multiple classes. In this case, the entropy threshold would let the detection pass because the background class has high confidence. Subsequently, a low per class confidence threshold does not restrict the boxes either. Therefore, the decoding output is worse than the actual predictions of the network. Bayesian SSD cannot help in this situation because the network is not actually uncertain.

SSD was developed with closed set conditions in mind. A well trained network in such a situation does not have many high confidence background detections. In an open set environment, however, background detections are the correct behaviour for unknown classes. In order to get useful detections out of the decoding, a higher confidence threshold is required.

3.3.2 Vanilla SSD with Entropy Thresholding

Vanilla SSD with entropy thresholding adds an additional component to the filtering already done for vanilla SSD. The entropy is calculated from all $\#nr\_classes$ softmax scores in a prediction. Only predictions with a low enough entropy pass the entropy threshold and move on to the aforementioned per class filtering. This excludes very uniform predictions but cannot identify false positive or false negative cases with high confidence values.

3.3.3 Bayesian SSD with Entropy Thresholding

Bayesian SSD uses multiple forward passes. Based on the information from Miller et al. [3], the detections of all forward passes are grouped per image but not by forward pass. This leads to the following shape of the network output after all forward passes: $(batch\_size, \#nr\_boxes \cdot \#nr\_forward\_passes, \#nr\_classes + 12)$. The size of the output increases linearly with more forward passes.

These detections have to be decoded first. Afterwards, all detections are thrown away which do not pass a confidence threshold for the class with the highest prediction probability. Additionally, all detections with a background prediction of 0.8 or higher are discarded. The remaining detections are partitioned into observations to further reduce the size of the output, and to identify uncertainty. This is accomplished by calculating the mutual IOU score of every detection with all other detections. Detections with a mutual IOU score of 0.95 or higher are partitioned into an observation. Next, the softmax scores and bounding box coordinates of all detections in an observation are averaged. There can be a different number of observations for every image, which destroys homogenity and prevents batch-wise calculation of the results. The shape of the results is per image: $(\#nr\_observations, \#nr\_classes + 4)$.

Entropy is measured in the next step. All observations with too high entropy are discarded. Entropy thresholding in combination with dropout sampling should improve identification of false positives of unknown classes. This is due to multiple forward passes and the assumption that uncertainty in some objects will result in different classifications in multiple forward passes. These varying classifications are averaged into multiple lower confidence values which should increase the entropy and, hence, flag an observation for removal.

The remainder of the filtering follows the vanilla SSD procedure: per class confidence threshold, NMS, and a top k selection at the end.

4 Experimental Setup and Results

This chapter explains the data sets used, and how the experiments have been set up. Furthermore, it presents the results.

4.1 Data Sets

This thesis uses the MS COCO [6] data set. It contains 80 classes, their range is illustrated by two classes: airplanes and toothbrushes. The images are real world images, ground truth is provided for all images. The data set supports object detection, keypoint detection, and panoptic segmentation (scene segmentation).

The data of any data set has to be prepared for use in a neural network. Typical problems of data sets include, for example, outliers and invalid bounding boxes. Before a data set can be used, these problems need to be resolved.

For the MS COCO data set, all annotations are checked for impossible values: bounding box height or width lower than zero, x_{min} and y_{min} bounding box coordinates lower than zero, x_{max} and y_{max} coordinates lower than or equal to zero, x_{min} greater than x_{max}, y_{min} greater than y_{max}, image width lower than x_{max}, and image height lower than y_{max}. In the last two cases the bounding box width and height are set to (image width - x_{min}) and (image height - y_{min}) respectively; in the other cases the annotation is skipped. If the bounding box width or height afterwards is lower than or equal to zero the annotation is skipped.

SSD accepts 300x300 input images, the MS COCO data set images are resized to this resolution; the aspect ratio is not kept in the process. MS COCO contains landscape and portrait images with (640x480) and (480x640) as the resolution. This leads to a uniform distortion of the portrait and landscape images respectively. Furthermore, the colour channels are swapped from RGB to BGR in order to comply with the SSD implementation. The BGR requirement stems from the usage of Open CV in SSD: the internal channel order for Open CV is BGR.

For this thesis, weights pre-trained on the sub data set trainval35k of the COCO data set are used. These weights have been created with closed set conditions in mind, therefore, they have been sub-sampled to create an open set condition. To

this end, the weights for the last 20 classes have been thrown away, making these classes, in effect, unknown.

All images of the minival2014 data set are used but only ground truth belonging to the first 60 classes is loaded. The remaining 20 classes are considered 'unknown' and no ground truth bounding boxes for them are provided during the inference phase. A total of 31,991 detections remain after this exclusion. Of these detections, a staggering 10,988 or 34,3% belong to the persons class, followed by cars with 1,932 or 6%, chairs with 1,791 or 5,6%, and bottles with 1,021 or 3,2%. Together, these four classes make up around 49,1% of the ground truth detections. This shows a huge imbalance between the classes in the data set.

4.2 Experimental Setup

This section explains the setup for the different conducted experiments. Each comparison investigates one particular question.

As a baseline, vanilla SSD with the confidence threshold of 0.01 and a NMS IOU threshold of 0.45 is used. Due to the low number of objects per image in the COCO data set, the top k value has been set to 20. Vanilla SSD with entropy thresholding uses the same parameters; compared to vanilla SSD without entropy thresholding, it showcases the relevance of entropy thresholding for vanilla SSD.

Vanilla SSD with 0.2 confidence threshold is compared to vanilla SSD with 0.01 confidence threshold; this comparison investigates the effect of the per class confidence threshold on the object detection performance.

Bayesian SSD with 0.2 confidence threshold is compared to vanilla SSD with 0.2 confidence threshold. Coupled with the entropy threshold, this comparison reveals how uncertain the network is. If it is very certain the dropout sampling should have no significant impact on the result. Furthermore, in two cases the dropout has been turned off to isolate the impact of NMS on the result.

Both vanilla SSD with entropy thresholding and Bayesian SSD with entropy thresholding are tested for entropy thresholds ranging from 0.1 to 2.4 inclusive as specified in Miller et al. [3].

4.3 Results

Results in this section are presented both for micro and macro averaging. In macro averaging, for example, the precision values of each class are added up and then divided by the number of classes. Conversely, for micro averaging the precision is calculated across all classes directly. Both methods have a specific impact: macro averaging weighs every class the same while micro averaging weighs every

detection the same. They will be largely identical when every class is balanced and has about the same number of detections. However, in case of a class imbalance the macro averaging favours classes with few detections whereas micro averaging benefits classes with many detections.

This section only presents the results. Interpretation and discussion is found in the next chapter.

4.3.1 Micro Averaging

	Forward Passes	max F_1 Score	abs OSE at max F_1 point	Recall at max F_1 point	Precision at max F_1 point
vanilla SSD - 0.01 conf		0.255	3176	0.214	0.318
vanilla SSD - 0.2 conf		**0.376**	2939	**0.382**	0.372
SSD with entropy test - 0.01 conf		0.255	3168	0.214	0.318
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.209	2709	0.300	0.161
no dropout - 0.2 conf - NMS	10	0.371	**2335**	0.365	**0.378**
0.9 keep ratio - 0.2 conf - NMS	10	0.359	2584	0.363	0.357
0.5 keep ratio - 0.2 conf - NMS	10	0.325	2759	0.342	0.311

Table 4.1: Rounded results for micro averaging. SSD with entropy test and Bayesian SSD are represented with their best performing entropy threshold with respect to F_1 score. Vanilla SSD with entropy test performed best with an entropy threshold of 2.4, Bayesian SSD without NMS performed best for 1.0, and Bayesian SSD with NMS performed best for 1.4 as entropy threshold. Bayesian SSD with dropout enabled and 0.9 keep ratio performed best for 1.4 as entropy threshold, the variant with 0.5 keep ratio performed best for 1.3 as threshold.

Vanilla SSD with a per class confidence threshold of 0.2 performs best (see table 4.1) with respect to the maximum F_1 score (0.376) and recall at the maximum F_1 point (0.382). In comparison, neither the vanilla SSD variant with a confidence threshold of 0.01 nor the SSD with an entropy test can outperform the 0.2 variant. Among the vanilla SSD variants, the 0.2 variant also has the lowest open set error (2939) and the highest precision (0.372).

The comparison of the vanilla SSD variants with a confidence threshold of 0.01 shows no significant impact of an entropy test. Only the open set error is lower but in an insignificant way. The rest of the performance metrics are identical after rounding.

Bayesian SSD with disabled dropout and without NMS has the worst performance of all tested variants (vanilla and Bayesian) with respect to F_1 score (0.209)

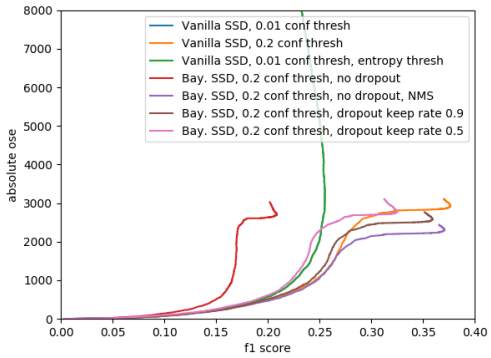


Figure 4.1: Micro averaged F_1 score versus open set error for each variant. Perfect performance is an F_1 score of 1 and an absolute OSE of 0.

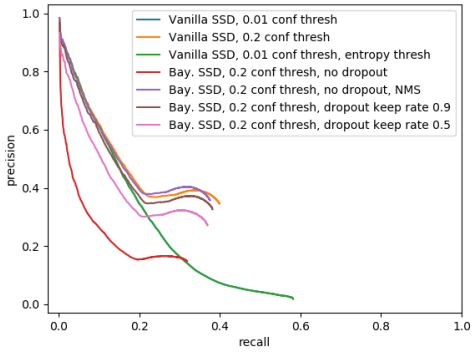


Figure 4.2: Micro averaged precision-recall curves for each variant tested.

and precision (0.161). The precision is not only the worst but also significantly lower compared to all other variants. In comparison to all variants with 0.2 confidence threshold, it has the worst recall (0.300) as well.

With an open set error of 2335, the Bayesian SSD variant with disabled dropout and enabled NMS offers the best performance with respect to the open set error. It also has the best precision (0.378) of all tested variants. Furthermore, it provides the best performance among all variants with multiple forward passes.

Dropout decreases the performance of the network, this can be seen in the lower F_1 scores, a higher open set error, and lower precision values. Both dropout variants have worse recall (0.363 and 0.342) than the variant with disabled dropout. However, all variants with multiple forward passes have a lower open set error than all vanilla SSD variants.

The relation of F_1 score to absolute open set error can be observed in figure 4.1. Precision-recall curves for all variants can be seen in figure 4.2. Both vanilla SSD variants with 0.01 confidence threshold reach a much higher open set error and a higher recall. This behaviour is to be expected as more and worse predictions are included. All plotted variants show a similar behaviour that is in line with previously reported figures, such as the ones in Miller et al. [3]

4.3.2 Macro Averaging

Vanilla SSD with a per class confidence threshold of 0.2 performs best (see table 4.2) with respect to the maximum F_1 score (0.375) and recall at the maximum F_1

	Forward Passes	max F_1 Score	abs OSE at max F_1 point	Recall at max F_1 point	Precision at max F_1 point
vanilla SSD - 0.01 conf		0.370	1426	0.328	0.424
vanilla SSD - 0.2 conf		**0.375**	1218	**0.338**	0.424
SSD with entropy test - 0.01 conf		0.370	1373	0.329	**0.425**
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.226	**809**	0.229	0.224
no dropout - 0.2 conf - NMS	10	0.363	1057	0.321	0.420
0.9 keep ratio - 0.2 conf - NMS	10	0.355	1137	0.320	0.399
0.5 keep ratio - 0.2 conf - NMS	10	0.322	1264	0.307	0.340

Table 4.2: Rounded results for macro averaging. SSD with entropy test and Bayesian SSD are represented with their best performing entropy threshold with respect to F_1 score. Vanilla SSD with entropy test performed best with an entropy threshold of 1.7, Bayesian SSD without NMS performed best for 1.5, and Bayesian SSD with NMS performed best for 1.5 as entropy threshold. Bayesian SSD with dropout enabled and 0.9 keep ratio performed best for 1.7 as entropy threshold, the variant with 0.5 keep ratio performed best for 2.0 as threshold.

point (0.338). In comparison, the SSD with an entropy test slightly outperforms the 0.2 variant with respect to precision (0.425). Additionally, this is the best precision overall. Among the vanilla SSD variants, the 0.2 variant also has the lowest open set error (1218).

The comparison of the vanilla SSD variants with a confidence threshold of 0.01 shows no significant impact of an entropy test. Only the open set error is lower but in an insignificant way. The rest of the performance metrics are almost identical after rounding.

The results for Bayesian SSD show a significant impact of NMS or the lack thereof: maximum F_1 score of 0.363 (with NMS) to 0.226 (without NMS). Dropout was disabled in both cases, making them, in effect, vanilla SSD with multiple forward passes.

With an open set error of 809, the Bayesian SSD variant with disabled dropout and without NMS offers the best performance with respect to the open set error. The variant without dropout and enabled NMS has the best F_1 score (0.363), the best precision (0.420) and the best recall (0.321) of all Bayesian variants.

Dropout decreases the performance of the network, this can be seen in the lower F_1 scores, a higher open set error, and lower precision and recall values. However, all variants with multiple forward passes have a lower open set error than all vanilla SSD variants.

The relation of F_1 score to absolute open set error can be observed in figure 4.3.

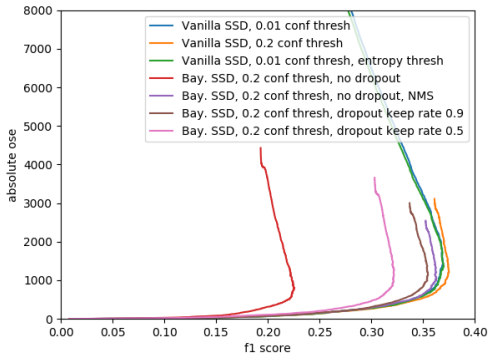


Figure 4.3: Macro averaged F_1 score versus open set error for each variant. Perfect performance is an F_1 score of 1 and an absolute OSE of 0.

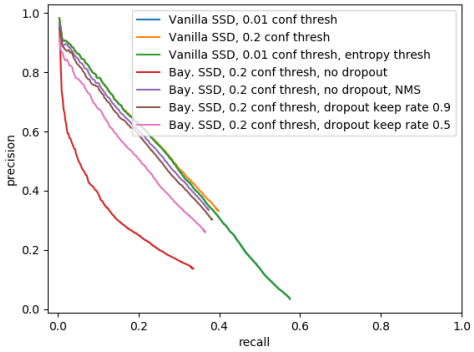


Figure 4.4: Macro averaged precision-recall curves for each variant tested.

Precision-recall curves for all variants can be seen in figure 4.4. Both vanilla SSD variants with 0.01 confidence threshold reach a much higher open set error and a higher recall. This behaviour is to be expected as more and worse predictions are included. All plotted variants show a similar behaviour that is in line with previously reported figures, such as the ones in Miller et al. [3]

4.3.3 Class-specific results

As mentioned before, the data set is imbalanced with respect to its classes: four classes make up roughly 50% of all ground truth detections. Therefore, it is interesting to see the performance of the tested variants with respect to these classes: persons, cars, chairs, and bottles. Additionally, the results of the giraffe class are presented as these are exceptionally good, although the class makes up only 0.7% of the ground truth. With this share, it is below the average of roughly 0.9% for each of the 56 classes that make up the second half of the ground truth.

In some cases, multiple variants have apparently the same performance but only one or some of them are marked bold. This is caused by differences prior to rounding: if two or more variants are marked bold they had the exact same performance before rounding.

The vanilla SSD variant with 0.2 per class confidence threshold performs best in the persons class: it has a max F_1 score of 0.460, consisting of a recall of 0.405 and a precision of 0.533. The variant shares the first place in recall with the vanilla SSD variant that uses a 0.01 confidence threshold. All Bayesian SSD variants

	Forward Passes	max F_1 Score	Recall at max F_1	Precision at max F_1 point
vanilla SSD - 0.01 conf		0.460	**0.405**	0.532
vanilla SSD - 0.2 conf		**0.460**	**0.405**	**0.533**
SSD with entropy test - 0.01 conf		0.460	0.405	0.532
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.272	0.292	0.256
no dropout - 0.2 conf - NMS	10	0.451	0.403	0.514
0.9 keep ratio - 0.2 conf - NMS	10	0.447	0.401	0.505
0.5 keep ratio - 0.2 conf - NMS	10	0.410	0.368	0.465

Table 4.3: Rounded results for persons class. SSD with entropy test and Bayesian SSD are represented with their best performing macro averaging entropy threshold with respect to F_1 score.

	Forward Passes	max F_1 Score	Recall at max F_1	Precision at max F_1 point
vanilla SSD - 0.01 conf		0.364	**0.305**	0.452
vanilla SSD - 0.2 conf		0.363	0.294	**0.476**
SSD with entropy test - 0.01 conf		**0.364**	**0.305**	0.453
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.236	0.244	0.229
no dropout - 0.2 conf - NMS	10	0.336	0.266	0.460
0.9 keep ratio - 0.2 conf - NMS	10	0.332	0.262	0.454
0.5 keep ratio - 0.2 conf - NMS	10	0.309	0.264	0.374

Table 4.4: Rounded results for cars class. SSD with entropy test and Bayesian SSD are represented with their best performing macro averaging entropy threshold with respect to F_1 score.

perform worse than the vanilla SSD variants (see table 4.3). With respect to the macro averaged result, all variants perform better than the average of all classes.

The performance for cars is slightly different (see table 4.4): the vanilla SSD variant with entropy threshold and 0.01 confidence threshold has the best F_1 score and recall. Vanilla SSD with 0.2 confidence threshold, however, has the best precision. Both the Bayesian SSD variant with NMS and disabled dropout, and the one with 0.9 keep ratio have a better precision (0.460 and 0.454 respectively) than the vanilla SSD variants with 0.01 confidence threshold (0.452 and 0.453). With respect to the macro averaged result, all variants have a better precision than the average. The Bayesian variant without NMS and dropout also has a better recall and F_1 score.

The best F_1 score (0.288) and recall (0.251) for the chairs class belongs to vanilla SSD with entropy threshold. Precision is mastered by Bayesian SSD with NMS

	Forward Passes	max F_1 Score	Recall at max F_1	Precision at max F_1 point
vanilla SSD - 0.01 conf		0.287	**0.251**	0.335
vanilla SSD - 0.2 conf		0.283	0.242	0.341
SSD with entropy test - 0.01 conf		**0.288**	**0.251**	0.338
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.172	0.168	0.178
no dropout - 0.2 conf - NMS	10	0.280	0.229	**0.360**
0.9 keep ratio - 0.2 conf - NMS	10	0.274	0.228	0.343
0.5 keep ratio - 0.2 conf - NMS	10	0.240	0.220	0.265

Table 4.5: Rounded results for chairs class. SSD with entropy test and Bayesian SSD are represented with their best performing macro averaging entropy threshold with respect to F_1 score.

	Forward Passes	max F_1 Score	Recall at max F_1	Precision at max F_1 point
vanilla SSD - 0.01 conf		0.233	**0.175**	0.348
vanilla SSD - 0.2 conf		0.231	0.173	**0.350**
SSD with entropy test - 0.01 conf		**0.233**	**0.175**	0.350
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.160	0.140	0.188
no dropout - 0.2 conf - NMS	10	0.224	0.170	0.328
0.9 keep ratio - 0.2 conf - NMS	10	0.220	0.170	0.311
0.5 keep ratio - 0.2 conf - NMS	10	0.202	0.172	0.245

Table 4.6: Rounded results for bottles class. SSD with entropy test and Bayesian SSD are represented with their best performing macro averaging entropy threshold with respect to F_1 score.

and disabled dropout (0.360). The variant with 0.9 keep ratio has the second-highest precision (0.343) of all variants. Both in F_1 score and recall, all Bayesian variants are worse than the vanilla variants. Compared with the macro averaged results, all variants perform worse than the average.

Bottles show similar performance to cars with overall lower numbers (see table 4.6). Again, all Bayesian variants are worse than all vanilla variants. The Bayesian SSD variant with NMS and disabled dropout has the best F_1 score (0.224) and precision (0.328) among the Bayesian variants; the variant with 0.5 keep ratio has the best recall (0.172). All variants perform worse than in the averaged results.

Last but not least the giraffe class (see table 4.7) is analysed. Remarkably, all three vanilla SSD variants have the identical performance, even before rounding. The Bayesian variant with NMS and disabled dropout outperforms all the other Bayesian variants with an F_1 score of 0.647, recall of 0.642, and 0.654 as precision.

	Forward Passes	max F_1 Score	Recall at max F_1	Precision at max F_1 point
vanilla SSD - 0.01 conf		**0.650**	**0.647**	**0.655**
vanilla SSD - 0.2 conf		**0.650**	**0.647**	**0.655**
SSD with entropy test - 0.01 conf		**0.650**	**0.647**	**0.655**
Bay. SSD - no DO - 0.2 conf - no NMS	10	0.415	0.414	0.417
no dropout - 0.2 conf - NMS	10	0.647	0.642	0.654
0.9 keep ratio - 0.2 conf - NMS	10	0.637	0.634	0.642
0.5 keep ratio - 0.2 conf - NMS	10	0.586	0.578	0.596

Table 4.7: Rounded results for giraffe class. SSD with entropy test and Bayesian SSD are represented with their best performing macro averaging entropy threshold with respect to F_1 score.

All variants perform better than in the macro averaged result.

4.3.4 Qualitative Analysis

This subsection compares vanilla SSD with Bayesian SSD with respect to specific images that illustrate similarities and differences between both approaches. For this comparison, a 0.2 confidence threshold is applied. Furthermore, the compared Bayesian SSD variant uses NMS and dropout with 0.9 keep ratio.

The ground truth only contains a stop sign and a truck. The differences between vanilla SSD and Bayesian SSD are almost not visible (see figures 4.5 and 4.6): the truck is neither detected by vanilla nor Bayesian SSD, instead both detected a 'potted plant' and a traffic light. The stop sign is detected by both variants. This behaviour implies problems with detecting objects at the edge that overwhelmingly lie outside the image frame. Furthermore, the predictions are usually identical.

Another example (see figures 4.7 and 4.8) is a cat with a laptop/TV in the background on the right side. Both variants detect a cat but the vanilla variant detects a dog as well. The laptop and TV are not detected but this is to be expected since these classes have not been trained.

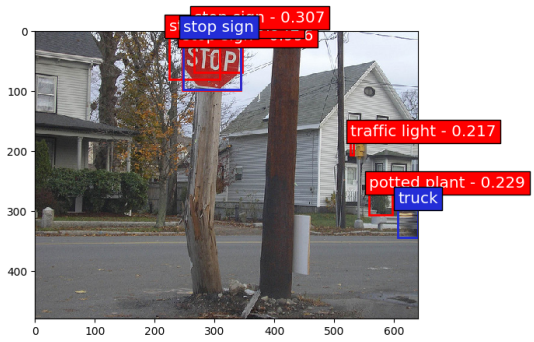


Figure 4.5: Image with stop sign and truck at right edge. Ground truth in blue, predictions in red, and rounded to three digits. Predictions are from vanilla SSD.

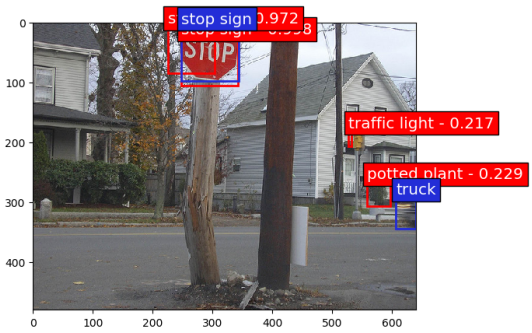


Figure 4.6: Image with stop sign and truck at right edge. Ground truth in blue, predictions in red, and rounded to three digits. Predictions are from Bayesian SSD with 0.9 keep ratio.

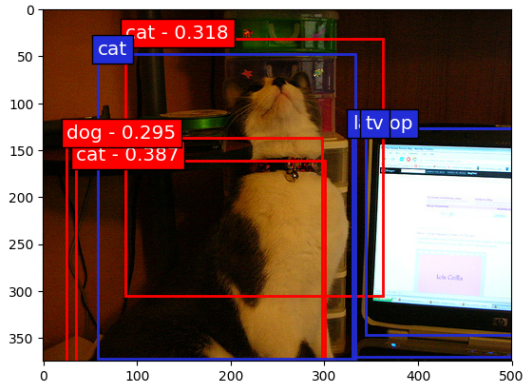


Figure 4.7: Image with a cat and laptop/TV. Ground truth in blue, predictions in red, and rounded to three digits. Predictions are from vanilla SSD.

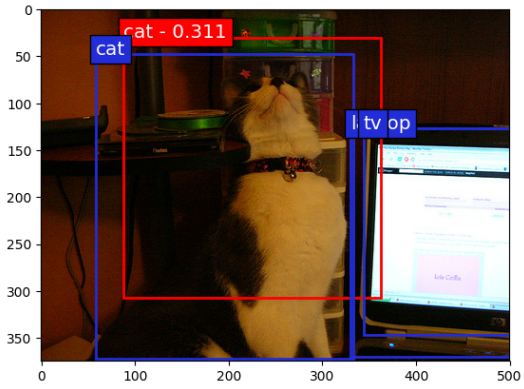


Figure 4.8: Image with a cat and laptop/TV. Ground truth in blue, predictions in red, and rounded to three digits. Predictions are from Bayesian SSD with 0.9 keep ratio.

5 Discussion and Outlook

First the results are discussed, then possible future research and open questions are addressed.

Discussion

The results clearly do not support the hypothesis: *Dropout sampling delivers better object detection performance under open set conditions compared to object detection without it.* With the exception of the open set error, there is no area where dropout sampling performs better than vanilla SSD. In the remainder of the section the individual results will be interpreted.

Impact of Averaging

Micro and macro averaging create largely similar results. Notably, micro averaging has a significant performance increase towards the end of the list of predictions. This is signaled by the near horizontal movement of the plot in both the F_1 versus absolute open set error graph (see figure 4.1) and the precision-recall curve (see figure 4.2).

This behaviour is caused by a large imbalance of detections between the classes. For vanilla SSD with 0.2 confidence threshold there are a total of 36,863 detections after NMS and top k. The persons class contributes 14,640 detections or around 40% to that number. Another strong class is cars with 2,252 detections or around 6%. In third place come chairs with 1352 detections or around 4%. This means that three classes have together roughly as many detections as the remaining 57 classes combined.

In macro averaging, the cumulative precision and recall values are calculated per class and then averaged across all classes. Smaller classes quickly reach high recall values as the total number of ground truth is small as well. The last recall and precision value of the smaller classes is repeated to achieve homogenity with the largest class. As a consequence, early on the average recall is quite high. Later on, only the values of the largest class still change which has only a small impact on the overall result.

variant	before entropy/NMS	after entropy/NMS	after top k
Bay. SSD, no dropout, no NMS	155,251	122,868	72,207
no dropout, NMS	155,250	36,061	33,827

Table 5.1: Comparison of Bayesian SSD variants without dropout with respect to the number of detections before the entropy threshold, after it and/or NMS, and after top k. The entropy threshold 1.5 was used for both.

Conversely, in micro averaging the cumulative true positives are added up across classes and then divided by the total number of ground truth. Here, the effect is the opposite: the total number of ground truth is very large which means the combined true positives of the 57 classes have only a smaller impact on the average recall. As a result, the open set error rises quicker than the F_1 score, creating the sharp rise of the open set error at a lower F_1 score than in macro averaging. The open set error reaches a high value early on and changes little afterwards. This allows the F_1 score to catch up and produces the almost horizontal line in the graph. Eventually, the F_1 score decreases again while the open set error continues to rise a bit.

Impact of Entropy

There is no visible impact of entropy thresholding on the object detection performance for vanilla SSD. This indicates that the network has almost no uniform or close to uniform predictions, the vast majority of predictions have a high confidence in one class—including the background. However, the entropy plays a larger role for the Bayesian variants—as expected: the best performing thresholds are 1.0, 1.3, and 1.4 for micro averaging, and 1.5, 1.7, and 2.0 for macro averaging. In all of these cases the best threshold is not the largest threshold tested.

This is caused by a simple phenomenon: at some point most or all true positives are in and a higher entropy threshold only adds more false positives. Such a behaviour is indicated by a stagnating recall for the higher entropy levels. For the low entropy thresholds, the low recall is dominating the F_1 score, the sweet spot is somewhere in the middle. For macro averaging, it holds that a higher optimal entropy threshold indicates a worse performance.

Non-Maximum Suppression and Top k

Miller et al. [3] supposedly do not use NMS in their implementation of dropout sampling. Therefore, a variant with disabled non-maximum suppression (NMS)

has been tested. The results are somewhat as expected: NMS removes all non-maximum detections that overlap with a maximum one. This reduces the number of multiple detections per ground truth bounding box and therefore the false positives. Without it, a lot more false positives remain and have a negative impact on precision. In combination with top k selection, recall can be affected: duplicate detections could stay and maxima boxes could be removed.

The number of observations have been measured before and after the combination of entropy threshold and NMS filter: both Bayesian SSD without NMS and dropout, and Bayesian SSD with NMS and disabled dropout have the same number of observations everywhere before the entropy threshold. After the entropy threshold (the value 1.5 has been used for both) and NMS, the variant with NMS has roughly 23% of its observations left (see table 5.1 for absolute numbers). Without NMS 79% of observations are left. Moreover, many classes have more observations after the entropy threshold and per class confidence threshold than before, which is clear since the background observations make up around 70% of the initial observations and only 21% of the initial observations are removed. Irrespective of the absolute number, this discrepancy clearly shows the impact of NMS and also explains a higher count of false positives: more than 50% of the original observations are removed with NMS and stay without—all of these are very likely to be false positives.

A clear distinction between micro and macro averaging can be observed: recall is hardly affected with micro averaging (0.300) but goes down noticeably with macro averaging (0.229). For micro averaging, it does not matter which class the true positives belong to: every detection counts the same way. This also means that top k will have only a marginal effect: some true positives might be removed without NMS but overall that does not have a big impact. With macro averaging, however, the class of the true positives matters a lot: for example, if two true positives are removed from a class with only few true positives to begin with than their removal will have a drastic influence on the class recall value and hence the overall result.

The impact of top k has been measured by counting the number of observations after top k is applied: the variant with NMS keeps about 94% of the observations left after NMS, without NMS only about 59% of observations are kept. This shows a significant impact on the result by top k in the case of disabled NMS. Furthermore, with disabled NMS some classes are hit harder by top k then others: for example, dogs keep around 82% of the observations but persons only 57%. This indicates that detected dogs are mostly on images with few detections overall and/or have a high enough prediction confidence to be kept by top k. However, persons are likely often on images with many detections and/or have too low confidences. In this example, the likelihood for true positives to be removed in

variant	after prediction	after observation grouping
Bay. SSD, no dropout, NMS	1,677,050	155,250
keep rate 0.9, NMS	1,617,675	549,166

Table 5.2: Comparison of Bayesian SSD variants without dropout and with 0.9 keep ratio of dropout with respect to the number of detections directly after the network predictions and after the observation grouping.

the person category is quite high. For dogs, the probability is far lower. This is a good example for micro and macro averaging, and their impact on recall.

Dropout Sampling and Observations

The dropout variants have largely worse performance than the Bayesian variants without dropout. This is to be expected as the network was not trained with dropout and the weights are not prepared for it.

Gal [18] shows that networks **trained** with dropout are approximate Bayesian models. The Bayesian variants of SSD implemented for this thesis are not fine-tuned or trained with dropout, therefore, they are not guaranteed to be such approximate models.

But dropout alone does not explain the difference in results. Both variants with and without dropout have the exact same number of detections coming out of the network (8732 per image per forward pass). With 16 images in a batch, 308 batches, and 10 forward passes, the total number of detections is an astounding 430,312,960 detections. As such a large number could not be handled in memory, only one batch is calculated at a time. That still leaves 1,397,120 detections per batch. These have to be grouped into observations, including a quadratic calculation of mutual IOU scores. Therefore, these detections are filtered by removing all those with background confidence levels of 0.8 or higher.

The number of detections per class has been measured before and after the detections are grouped into observations. To this end, the stored predictions are unbatched and summed together. After the aforementioned filter and before the grouping, roughly 0.4% (in fact less than that) of the more than 430 million detections remain (see table 5.2 for absolute numbers). The variant with dropout has slightly fewer predictions left compared to the one without dropout.

After the grouping, the variant without dropout has on average between 10 and 11 detections grouped into an observation. This is to be expected as every forward pass creates the exact same result and these ten identical detections per vanilla SSD detection perfectly overlap. The fact that slightly more than ten detections

are grouped together could explain the marginally better precision of the Bayesian variant without dropout compared to vanilla SSD. However, on average only three detections are grouped together into an observation if dropout with 0.9 keep ratio is enabled. This does not negatively impact recall as true positives do not disappear but offers a higher chance of false positives. It can be observed in the results which clearly show no negative impact for recall between the variants without dropout and dropout with 0.9 keep ratio.

This behaviour implies that even a slight usage of dropout creates such diverging anchor box offsets that the resulting detections from multiple forward passes no longer have a mutual IOU score of 0.95 or higher.

Outlook

The attempted replication of the work of Miller et al. raises a series of questions that cannot be answered in this thesis. This thesis offers one possible implementation of dropout sampling that technically works. However, this thesis cannot answer why this implementation differs significantly from Miller et al. The complete source code or otherwise exhaustive implementation details of Miller et al. would be required to attempt an answer.

Future work could explore the performance of this implementation when used on an SSD variant that was fine-tuned or trained with dropout. In this case, it should also look into the impact of training with both dropout and batch normalisation. Other avenues include the application to other data sets or object detection networks.

To facilitate future work based on this thesis, the source code will be made available and an installable Python package will be uploaded to the PyPi package index. More details about the source code implementation and additional figures can be found in the appendices.

Bibliography

[1] B. Friedman and H. Nissenbaum, 'Bias in computer systems', *ACM Transactions on Information Systems*, vol. 14, no. 3, pp. 330–347, 1996. DOI: 10.1145/230538.230561 (cit. on p. 1).

[2] N. Diakopoulos, 'Algorithmic accountability reporting: On the investigation of black boxes', Tow Center for Digital Journalism, Columbia University, research rep., 2014. DOI: 10.7916/d8zk5tw2 (cit. on p. 1).

[3] D. Miller *et al.*, 'Dropout sampling for robust object detection in open-set conditions', in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2018. DOI: 10.1109/icra.2018.8460700 (cit. on pp. 2, 3, 6, 7, 9, 11, 13, 15, 17, 19, 25).

[4] M. A. F. Pimentel *et al.*, 'A review of novelty detection', *Signal Processing*, vol. 99, pp. 215–249, 2014 (cit. on pp. 3, 5, 6).

[5] L. Deng, 'The MNIST database of handwritten digit images for machine learning research [best of the web]', *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012. DOI: 10.1109/msp.2012.2211477 (cit. on p. 3).

[6] T.-Y. Lin *et al.*, 'Microsoft COCO: Common objects in context', in *Computer Vision – ECCV 2014*, Springer International Publishing, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48 (cit. on pp. 3, 14).

[7] J. McCormac *et al.*, 'SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation?', in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017. DOI: 10.1109/iccv.2017.292 (cit. on p. 3).

[8] S. Pidhorskyi *et al.*, 'Generative probabilistic novelty detection with adversarial autoencoders', in *Advances in Neural Information Processing Systems*, S. Bengio *et al.*, Eds., vol. 31, Curran Associates, Curran Associates, Inc., 2018, pp. 6823–6834 (cit. on p. 5).

[9] V. Hautamaki *et al.*, 'Outlier detection using k-nearest neighbour graph', in *Proceedings of the 17th International Conference on Pattern Recognition, 2004*, IEEE, 2004. DOI: 10.1109/icpr.2004.1334558 (cit. on p. 5).

[10] M. I. Jordan and R. A. Jacobs, 'Hierarchical mixtures of experts and the EM algorithm', *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994. DOI: 10.1162/neco.1994.6.2.181 (cit. on p. 5).

[11] Q. Song *et al.*, 'Robust support vector machine with bullet hole image classification', *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 4, pp. 440–448, 2002. DOI: 10.1109/tsmcc.2002.807277 (cit. on p. 5).

[12] M. Filippone and G. Sanguinetti, 'A perturbative approach to novelty detection in autoregressive models', *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 1027–1036, Mar. 2011. DOI: 10.1109/tsp.2010.2094609 (cit. on p. 6).

[13] C. M. Bishop, 'Novelty detection and neural network validation', *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 141, no. 4, pp. 217–222, 1994. DOI: 10.1049/ip-vis:19941330 (cit. on p. 6).

[14] Z. Ghahramani, 'Probabilistic machine learning and artificial intelligence', *Nature*, vol. 521, no. 7553, pp. 452–459, 2015. DOI: 10.1038/nature14541 (cit. on p. 6).

[15] D. J. C. MacKay, 'A practical bayesian framework for backpropagation networks', *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992. DOI: 10.1162/neco.1992.4.3.448 (cit. on p. 6).

[16] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer New York, 1996. DOI: 10.1007/978-1-4612-0745-0 (cit. on p. 6).

[17] Y. Gal and Z. Ghahramani, 'Dropout as a bayesian approximation: Representing model uncertainty in deep learning', in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 20th–22nd Jun. 2016, pp. 1050–1059 (cit. on p. 6).

[18] Y. Gal, 'Uncertainty in deep learning', PhD thesis, University of Cambridge, 2017 (cit. on pp. 6, 11, 27).

[19] D. Miller *et al.*, 'Evaluating merging strategies for sampling-based uncertainty techniques in object detection', *arXiv preprint*, 17th Sep. 2018. arXiv: 1809.06006v3 [cs.CV] (cit. on p. 6).

[20] M. Teye *et al.*, 'Bayesian uncertainty estimation for batch normalized deep networks', *arXiv preprint*, 18th Feb. 2018. arXiv: 1802.06455v2 [stat.ML] (cit. on p. 6).

[21] S. Ioffe and C. Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint*, 2nd Mar. 2015. arXiv: 1502.03167v3 [cs.LG] (cit. on p. 6).

[22] X. Li *et al.*, 'Understanding the disharmony between dropout and batch normalization by variance shift', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2682–2690 (cit. on p. 7).

[23] J. Postels *et al.*, 'Sampling-free epistemic uncertainty estimation using approximated variance propagation', *arXiv preprint*, 21st Aug. 2019. arXiv: 1908.00598v2 [cs.LG] (cit. on p. 7).

[24] B. Lakshminarayanan *et al.*, 'Simple and scalable predictive uncertainty estimation using deep ensembles', in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017, pp. 6402–6413 (cit. on p. 7).

[25] Y. Geifman *et al.*, 'Bias-reduced uncertainty estimation for deep neural classifiers', *arXiv preprint*, 30th Sep. 2018. arXiv: 1805.08206v3 [cs.LG] (cit. on p. 7).

[26] M. Sensoy *et al.*, 'Evidential deep learning to quantify classification uncertainty', in *Advances in Neural Information Processing Systems*, S. Bengio *et al.*, Eds., vol. 31, Curran Associates, Inc., 2018, pp. 3183–3193. [Online]. Available: http://papers.nips.cc/paper/7580-evidential-deep-learning-to-quantify-classification-uncertainty.pdf (cit. on p. 7).

[27] J. Mukhoti and Y. Gal, 'Evaluating bayesian deep learning methods for semantic segmentation', *arXiv preprint*, 30th Nov. 2018. arXiv: 1811.12709v1 [cs.CV] (cit. on p. 7).

[28] A. Wu *et al.*, 'Deterministic variational inference for robust bayesian neural networks', in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1l08oAct7 (visited on 03/01/2019) (cit. on p. 7).

[29] A. Kendall and Y. Gal, 'What uncertainties do we need in bayesian deep learning for computer vision?', in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017, pp. 5574–5584 (cit. on p. 8).

[30] W. Liu *et al.*, 'SSD: Single shot MultiBox detector', in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2 (cit. on pp. 9, 10, 12).

[31] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/ (cit. on p. 11).

[32] F. Chollet *et al.*, *Keras*, `https://keras.io`, 2015 (cit. on p. 11).

Appendix A

Software and Source Code Design

The source code of many published papers is either not available or is of bad quality: it is poorly documented, difficult to integrate into your own work, and often does not follow common software development best practices. Moreover, with Tensorflow, PyTorch, and Caffe there are at least three machine learning frameworks. Every research team seems to prefer another framework, and, occasionally, even develops their own; this makes it difficult to combine the work of different authors. In addition to this, most papers do not contain proper information regarding implementation details, making it difficult to accurately replicate their results, if their source code is not available.

Therefore, I will release my source code and make it available as a Python package on the PyPi package index. This makes it possible for other researchers to simply install a package and use the API to interact with my code. Additionally, the code has been designed to be future proof, and work with the announced Tensorflow 2.0, by supporting eager mode.

Furthermore, it is configurable, well documented, and conforms largely to the clean code guidelines: evolvability and extendability among others.

The code was designed to be modular: One module creates the command line interface (main.py), another implements the actions chosen in the CLI (cli.py), the MS COCO to SceneNet RGB-D mapping can be found in the definitions.py module, preparation of the data sets and retrieval of data is grouped in the data.py module, evaluation metrics have their separate module (evaluation.py), the configuration is accessed and handled by the config.py module, plotting-related code can be found in plotting.py, and the ssd.py module contains code to train the SSD and later predict with it.

Lastly, the SSD implementation from a third party repository has been modified to work inside a Python package architecture, and with eager mode. It is stored as a Git submodule inside the package repository.

Appendix B

More Figures and Data

In this chapter you can find further tables that back up some of the points in the thesis but did not make it into it.

class	number of detections	percentage of total
total	31,991	100%
persons	10,988	34.3%
cars	1,932	6%
chairs	1,791	5.6%
bottles	1,021	3.2%
cups	898	2.8%

Table B.1: Number of ground truth detections per class (top 5).

variant	before entropy/NMS	after entropy/NMS	after top k
Bay. SSD, no dropout, no NMS	19,014	48,484	27,707
no dropout, NMS	19,014	14,542	13,486

Table B.2: Comparison of Bayesian SSD variants without dropout with respect to the number of detections before the entropy threshold, after it and/or NMS, and after top k. The entropy threshold 1.5 was used for both. The numbers are for the persons class.

variant	before entropy/NMS	after entropy/NMS	after top k
Bay. SSD, no dropout, no NMS	1,011	1,785	1,458
no dropout, NMS	1,011	426	425

Table B.3: Comparison of Bayesian SSD variants without dropout with respect to the number of detections before the entropy threshold, after it and/or NMS, and after top k. The entropy threshold 1.5 was used for both. The numbers are for the dogs class.

Glossary

BGR stands for the three colour channels blue, green, and red in this order . 14

Caffe is a deep learning framework written in C++ . 12

CCTV stands for closed-circuit television or video surveillance . 1

Dirichlet distribution is named after Peter Dirichlet and a family of probability distributions . 7

entropy describes the amount of information provided by something. More likely events have a lower entropy than rare events. In case of classification probabilities, uniform predictions contain more information than predictions with a clear "winner" . 3, 5, 8, 9, 11–13, 15, 16, 18, 20–22, 25, 26, 34, 35

Hopfield network is a recurrent neural network. Used as "associative" memory systems with binary thresholds. Guaranteed to converge to local minimum, this can be the wrong one though . 6

MCBN Monte Carlo Batch Normalisation. 7

MCDO Monte Carlo Dropout. 6, 7

MLP multilayer perceptron. 6

NMS non-maximum suppression. 3, 12, 13, 15–18, 20–22, 24–27, 34, 35

OSE open set error. 3, 17, 19

pdf probabilistic density function. 5

posterior probability output of a neural network . 7, 8

RGB stands for the three colour channels red, green, and blue in this order . 14

SSD Single Shot MultiBox Detector. 3, 4, 9–28, 34, 35

vanilla is used to describe the original state of something . 3, 4, 10–13, 15–25, 27, 28

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Master Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zu Bibliothek

Ich bin damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek eingestellt wird.

Ort, Datum Unterschrift