Université de Montréal

Unknown object identification in road scenarios in autonomous-driving: OOD detection in object detection vision systems

par

Vaibhav Sanjay Jade

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Rapport de stage présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Apprentissage automatique

June 26, 2024

© Vaibhav Sanjay Jade, 2023

Abstract

In this work, we investigate unknown object detection in vision systems, specifically for autonomous driving. Detecting unknown objects during object detector training, and solving overconfidence issues of model prediction are crucial for road safety and reliability of autonomous driving systems. We define both localizing unseen objects and evaluating the confidence of model predictions as this study's two central task objectives.

First, we understand the latent representation space using intermediate activations generated through our Faster-RCNN model. We evaluate these latent spaces at various depths of representation of activations and visualize the clusters formed using t-SNE, extracted from ground truth annotations of training samples. As we employ the Mahalanobis distance-based metric for OOD evaluation, we also study cluster separability using SVM decision boundaries.

We first propose our algorithm utilizing the RPN's (subnetwork of the Faster-RCNN model) 'objectness' bounding box predictions. We use pre-trained RPN from open-world large-scale datasets to generate object proposals for unseen objects w.r.t. our custom in-distribution dataset. We then evaluate and report this model's performance on the custom dataset.

In order to perform OOD unknown object detection using these fine-tuned models, We modify the original algorithm proposed in [**1**] for OOD detection. We adapt it from classification literature to object detection, which involves simultaneous localization and classification. We perform unknown object detection using the region proposals from the RPN and evaluate confidence scores using the modified strategy (detailed in section 3.6). We further improve the algorithm by using predictions from both pre-trained as well as fine-tuned models as proposals for OOD object evaluation.

Contents

Abstract 2

List of tables 6

List of figures 7

List of acronyms and abbreviations 10

Acknowledgements 12

Chapter 1. OOD detection And Related Work 13

1.1. Introduction 13

1.2. Out-of-Distribution(OOD) Classification 14

1.2.1. Background 14

1.2.2. Definition[**2**] 14

1.2.3. Types of methods 14

1.2.4. LogitNorm (Classification-based) 15

1.2.5. Mahalanobis distance (Distance-based) 16

1.3. Object Detection 17

1.3.1. Two-stage detectors 17

1.3.2. Single-shot detectors 18

1.3.3. Anchor-free and Transformers 18

1.4. Unknown-Aware Object Detection 19

1.4.1. Unknown Distillation 19

1.4.2. Unknown candidate selection 19

1.4.3. Uncertainty Regularization 20

Chapter 2. Dataset 21

2.1. MS COCO dataset [**3**] 21

2.2. BigDetection dataset [**4**] 22

2.3. E-smart custom dataset 24

2.4. Visualizing the object clusters: E-Smart dataset 26
2.4.1. Methodology 26
2.4.2. How to generate feature embeddings for a given bounding box? 28
2.4.3. MS COCO pretrained 29
2.4.3.1. ROI pooling layer (7x7x256 dim) 29
2.4.3.2. Penultimate layer (1024 dim) 30
2.4.3.3. Linear separability 31
2.4.4. BigDetection pretrained 32
2.4.4.1. ROI pooling layer (7x7x256 dim) 32
2.4.4.2. Penultimate layer (1024 dim) 32
2.4.4.3. Linear separability test 33
2.4.5. Conclusion 34

Chapter 3. Methodology and Model Pipeline 35

3.1. Understanding OOD samples in object detection 35

3.2. Faster RCNN model architecture 36
3.2.1. Feature Pyramid Network 37
3.2.2. Region Proposal Network 38
3.2.3. Box Head 40

3.3. Why do we need pre-trained RPN for OOD detection? 42

3.4. Fine-tuning on in-distribution dataset 42
3.4.1. Results of fine-tuning on E-smart 42
3.4.1.1. Visualizing the results on OOD samples 44

3.5. OOD detection with Mahalanobis distance 47

3.6. Adapting Mahalanobis distance for the Faster-RCNN 48
3.6.1. Using pre-trained model predictions instead of region proposals 50

3.7. Results 53
3.7.1. Issues with just using region proposals 53
3.7.2. Visualising the OOD Detection 55

Chapter 4. Conclusion 58

4.1. Discussion 58

4.2. Future work 59

References 60

List of tables

2.1 The class distribution of bounding boxes in E-Smart dataset 25
2.2 Training Metrics for MS COCO pre-trained model 31
2.3 Training Metrics for BigDetection pre-trained model 33

3.1 The results of COCO evaluation metric of finetuning on E-Smart dataset, using MS COCO pre-trained model. The highlighted rows depict the main metric as defined by the COCO challenge. 43
3.2 The results of COCO evaluation metric of finetuning on E-Smart dataset, using BigDetection pre-trained model. The highlighted rows depict the main metric as defined by the COCO challenge. 44

List of figures

1.1 **(a)** Softmax outputs of 2 examples of CIFAR-10 trained model **(b)** TSNE visualization of softmax output on CIFAR-10 trained model 16

1.2 The overview of STUD architecture[**5**] 20

2.1 MS COCO dataset class list[**3**] 22

2.2 (On left)The category mapping pipeline for label-space generation, demonstrating the merging process of semantic labels using BERT (On right) The statistics of individual datasets and after merging into BigDetection (Source: paper[**4**]) 23

2.3 Histogram of class-distribution in the E-Smart dataset. 25

2.4 The architectural diagram representing the algorithm to compute feature vectors for a given image and corresponding ground truth bounding boxes. Black signifies the traditional forward pass and blue signifies the modified pass-through. Note that the ROI pooler uses ground truth boxes instead of RPN proposals here. 27

2.5 The architectural diagram representing the algorithm to compute feature vectors for a given image and corresponding predicted bounding boxes. Black signifies the traditional forward pass and blue signifies the modified pass-through. Note that in the second pass, the ROI pooler uses ground truth boxes instead of RPN proposals to generate the feature vectors. 27

2.6 The t-SNE visualization of feature vectors of ROI pooling layer after PCA dimensionality reduction to 1024 dims. 29

2.7 The t-SNE visualization of feature vectors of penultimate layer, outputs are of 1024 dims. 30

2.8 The t-SNE visualization of feature vectors of ROI pooling layer after PCA, outputs are of 1024 dims. 32

2.9 The t-SNE visualization of feature vectors of ROI pooling layer after PCA, outputs are of 1024 dims. 33

3.1 Detailed architecture of Base-RCNN-FPN, blue signifies class names, source[**6**] . . 36

3.2 Detailed architecture of Feature pyramid network with ResNet backbone. Blue labels are individual blocks in the architecture. (a),(b), (c) are different types of bottleneck blocks in ResNet.[**6**] 37

3.3 Visualizing the feature map at each level from P2 to P6, for the given input image. These are 0th index channels.[**6**] 38

3.4 Detailed architecture of Region Proposal Network.[**6**] 39

3.5 Visualization of objectness map on the input image, for each feature map level. Sigmoid is applied to objectness logits.[**6**] 39

3.6 Detailed architecture of Box Head.[**6**] 40

3.7 Feature level assignment for ROI pooling. Here, the grey and blue boxes are assigned to levels P2 and P5 based on area heuristic, respectively.[**6**] 41

3.8 a) Elephant crossing road b) MS COCO output c) BigDetection output 44

3.9 a) Yak crossing road b) MS COCO output c) BigDetection output 45

3.10 a) Deer crossing road b) MS COCO output c) Big Detection output 45

3.11 a) Bad visibility on road b) MS COCO output c) BigDetection output 45

3.12 a) Cars on snowy road b) MS COCO output c) BigDetection output 45

3.13 a) Road under construction b) MS COCO output c) BigDetection output 46

3.14 Visualizing the results of the fine-tuned models on images containing OOD samples 46

3.15 Schematic for calculating the gradient image w.r.t. input using Mahalanobis distance loss per proposed bounding box. Black signifies the traditional forward pass, red signifies the backdrop computational flow. 50

3.16 Schematic for computing the final confidence score based on Mahalanobis distance, using copy model and given bounding box. Black signifies the traditional forward pass, blue denotes the second pass-through for calculating the Gaussian score. Note that RPN and box head from the traditional forward pass are blurred (and are italics) as they are not used. 51

3.17 Schematic for computing the final confidence score based on Mahalanobis distance, using copy model and given bounding box. Black signifies the traditional forward pass, blue denotes the second pass-through for calculating the Gaussian score. Red denotes the gradient flow to the input image. Note that RPN from the

traditional forward pass is blurred (and in italics) as they are not used in gradient computation. 52
3.18 (a)Output of region proposals 53
3.19 (b)Regions after NMS 53
3.20 (c)Output of OOD detection 53
3.21 (a)Output of region proposals 54
3.22 (b)Regions after NMS 54
3.23 (c)Output of OOD detection 54
3.24 a) Image before OOD detection b) MS COCO output c) BigDetection output . . . 55
3.25 a) Image before OOD detection b) MS COCO output c) BigDetection output . . . 55
3.26 a) Image before OOD detection b) MS COCO output c) BigDetection output . . . 56
3.27 a) Image before OOD detection b) MS COCO output c) BigDetection output . . . 56
3.28 a) Image before OOD detection b) MS COCO output c) BigDetection output . . . 56
3.29 a) Image before OOD detection b) MS COCO output c) BigDetection output . . . 57
3.30 Qualitative assessment of pre-trained models with OOD detection 57

List of acronyms and abbreviations

OOD	Out-of-Distribution
ID	In-Distribution
FPN	Feature Pyramid Network
RPN	Region Proposal Network
PCA	Principal Component Anaysis
t-SNE	t-distributed Stochastic Neighbor Embedding
IoU	Intersection over Union
CNN	Convolutional Neural Network
ROI	Region Of Interest
SVM	Support Vector Machine

GDA	Gaussian discriminant analysis
RCNN	Region-based convolution neural network
YOLO	You-Only Look Once
SSD	Single-Shot multibox Detector
FCOS	Fully Convolutional One-Stage object detection
DETR	DEtection TRansformer
STUD	Spatial-Temporal Unknown Distillation
GAN	Generative Adversarial Networks
MS COCO	Microsoft Common Objects in Context
LVIS	Large Vocabulary Instance Segmentation
BERT	Bidirectional Encoder Representations from Transformers

Acknowledgements

I would like to sincerely thank everyone who helped to see this project through to its successful conclusion.

First and foremost, I value the facilities and resources that E-smart Control supplied, as they were crucial to the project's success. Moreover, I would like to express my sincere gratitude to my manager, Raphaël Jean, for his invaluable assistance and insightful advice, which were central in setting up the research direction. I would also like to thank my mentor, Stefan Horoi, whose advice, regular discussions, and feedback were extremely helpful throughout the entire project. Additionally, I would like to thank Mitacs for their financial support. Finally, I want to express my gratitude to my friends and family for their continuous support and patience throughout this endeavor.

Chapter 1

OOD detection And Related Work

1.1. Introduction

From the introduction of "Transfer learning" in deep learning models, such as from large-scale dataset ImageNet for computer vision, the study of learning algorithms for distribution shift, domain adaptation, few-shot/zero-shot learning, and OOD detection has become central research topics in deep learning. With diverse applications across various domains such as autonomous driving, warehouse automation, satellite imagery, medical imaging, etc. understanding the predictions of otherwise black-box models also flourished the interpretability and explainability research domain.

The bedrock of machine learning models lies in the closed-world assumption, presupposing that test data follows the same distribution as the training dataset, categorized as in-distribution (ID). However, as these models step into open-world scenarios, the emergence of out-of-distribution (OOD) test samples necessitates cautious handling. These distributional shifts stem from either semantic discrepancies, where OOD samples originate from distinct classes, or covariate shifts, where the OOD samples come from different domains. Two distinct research communities have emerged to address the challenges posed by OOD samples. The OOD generalization focuses on fortifying models against domain shift, striving to maintain robust performance on OOD samples. Conversely, OOD detection prioritizes model reliability by specifically identifying samples exhibiting semantic shifts, where the model either cannot or should not generalize. [**7**]

1.2. Out-of-Distribution(OOD) Classification

1.2.1. Background

The emergence of the out-of-distribution detection field arose from the realization that deep learning models exhibit overconfidence when classifying samples stemming from diverse semantic distributions in tasks like image classification and text categorization. Consequently, the impetus for OOD detection grew, necessitating models to discern and reject inputs that diverge semantically from the training distribution and should not be predicted.

1.2.2. Definition[**2**]

At its core, out-of-distribution detection seeks to identify test samples drawn from distributions distinct from the training distribution, with the specifics of distribution defined based on the application context. Typically, in most machine learning tasks, the distribution refers to the "label distribution," signifying that *OOD samples should not have overlapping labels with the training data.* Formally, in OOD detection, test samples originate from a distribution whose semantics differ from the ID, i.e., $P(Y) \neq P'(Y)$. It's crucial to note that while the training set encompasses multiple classes, OOD detection should not compromise the ID classification capability.

1.2.3. Types of methods

Over the past half-decade, a rich assortment of methodologies has been developed for OOD detection, which can be broadly classified into categories such as classification-based, density-based, and distance-based methods [**2**].

(1) Classification-based techniques, constituting a major portion of OOD detectors, derive confidence directly from the classifier, often incorporating specific design modifications tailored for effective OOD detection. These modifications can revolve around alterations in loss functions, classifier architectures, or post-hoc processing techniques.

(2) Density-based methods, on the other hand, model the in-distribution using probabilistic models, offering strong performance and facilitating theoretical analysis.

(3) Meanwhile, distance-based methods leverage computations in high-dimensional spaces like feature or gradient spaces to delineate between ID and OOD samples.

(4) Additionally, there exist minor categories like reconstruction-based methods, relying on discrepancies in reconstruction errors between ID and OOD samples to detect deviations from the training distribution.

1.2.4. LogitNorm (Classification-based)

Paper [**8**] introduces Logit Normalization (LogitNorm) as an uncomplicated yet potent alternative to the cross-entropy loss, amplifying numerous existing post-hoc methods tailored for OOD detection. By disentangling the influence of logits' norm from the training objective, LogitNorm fosters a tendency for conservative predictions on OOD inputs, enhancing the separation between OOD and in-distribution (ID) data. This method decouples the influence of output norm during network optimization, curbing the escalating logit norm observed during training and consequently mitigating overconfident output.

LogitNorm, maintains consistency between the direction of the logit and its corresponding one-hot label, disregarding the logit's magnitude during optimization. This method involves normalizing the logit vector to a unit vector with a constant magnitude, followed by applying softmax cross-entropy loss on this normalized logit vector instead of the original output. Formally, the objective function of LogitNorm is given by:

$$R_L(f) = \mathbb{E}_{(x,y)\sim P_{XY}} \left[\mathcal{L}_{CE} \left(\hat{f}(x;\theta), y \right) \right]$$

where $\hat{f}(x;\theta) = \frac{f(x;\theta)}{\| f(x;\theta) \|}$is the normalized logit vector. Equivalently, the new loss function can be defined as:

$$L_{\text{logit norm}}(f(x;\theta), y) = -\log \frac{e^{f_y/(\tau\|f\|)}}{\sum_{i=1}^{k} e^{f_i/(\tau\|f\|)}}$$

This loss function can be viewed as having an input-dependent temperature, $\tau\|f(x;\theta)\|$, which depends on the input x. Due to the formulation of logit normalization, the magnitudes of the output vectors are strictly constant (i.e., $1/\tau$). Minimizing the loss can only be achieved by adjusting the direction of the logit output f. The resulting model tends to give conservative predictions, especially for inputs far away from P_{in}.

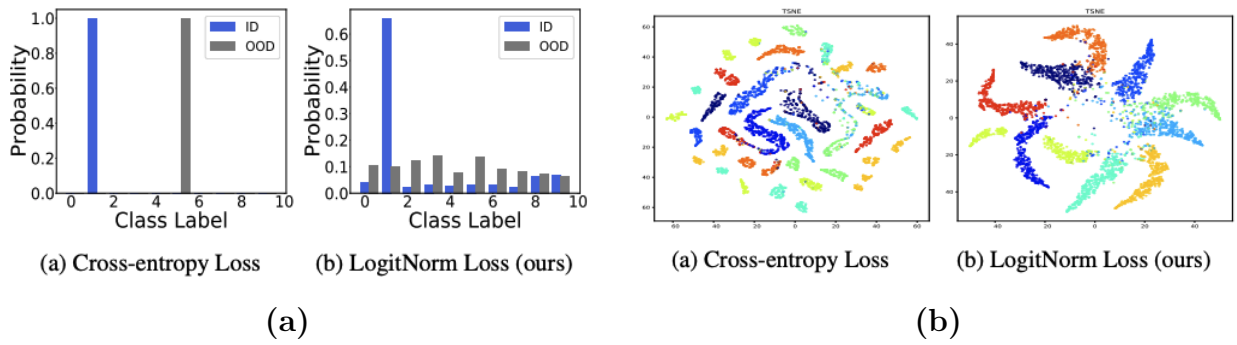


Fig. 1.1. **(a)** Softmax outputs of 2 examples of CIFAR-10 trained model
(b) TSNE visualization of softmax output on CIFAR-10 trained model

1.2.5. Mahalanobis distance (Distance-based)

This paper [**1**] introduces a versatile method capable of effectively detecting abnormal samples applicable to any pre-trained softmax neural classifier. By leveraging Gaussian discriminant analysis on deep model features, it derive class conditional Gaussian distributions, enabling the computation of confidence scores through the Mahalanobis distance.

The generative classifier's posterior distribution under GDA, assuming tied covariance, mirrors the behavior of the softmax classifier. Consequently, the features derived from the pre-trained softmax neural classifier adhere to class-conditional Gaussian distributions.

For parameter estimation of this generative classifier, empirical class means and covariances are computed from the training samples:

$$\mu_{bc} = \frac{1}{N_c} \sum_{i:y_i=c} f(x_i), \quad \Sigma_b = \frac{1}{N} \sum_c \sum_{i:y_i=c} (f(x_i) - \mu_{bc})(f(x_i) - \mu_{bc})^T,$$

where N_c denotes the count of training samples with label c.

This process aligns with fitting class-conditional Gaussian distributions with a tied covariance to the training samples using the maximum likelihood estimator, illustrating a unique convergence between GDA and the softmax classifier. This allows for the effective utilization of pre-trained model features for abnormality detection without necessitating alterations to the classifier's architecture.

Initially, the softmax classifier's outputs are leveraged to establish class conditional probabilities using the softmax function:

$$P(y = c|x) = \frac{\exp(w_c^T f(x) + b_c)}{\sum_{c' \neq c} \exp(w_{c'}^T f(x) + b_{c'})},$$

where w_c and b_c represent the weight and bias of class c, and $f(\cdot)$ denotes the penultimate layer's output of the neural network.

This information feeds into a generative classifier that assumes a multivariate Gaussian distribution for class-conditional distributions:

$$P(f(x)|y = c) = \mathcal{N}(f(x)|\mu_c, \Sigma),$$

where μ_c signifies the mean for class c in the Gaussian distribution, and Σ denotes the tied covariance.

Unlike prior methods specialized in detecting either out-of-distribution or adversarial samples, this approach achieves state-of-the-art performance in both detecting OOD and adversarial examples, demonstrating superior adaptability and robustness, particularly in challenging conditions like datasets with noisy labels or limited samples.

1.3. Object Detection

The methods introduced in the object detection literature can be broadly divided into two classes: Sliding window-based methods (Single stage) and region proposal classification (Two stage). With the advent of convolutional neural networks and deep learning, the traditional methods in both classes further integrated neural network-based object detection. With the recent introduction of the attention mechanism, transformer architectures are also adapted for object detection.

1.3.1. Two-stage detectors

For region proposal classification class, R-CNN [**9**] introduced a CNN-based post-classification on selective search-based region proposals. Fast-RCNN [**10**] introduced ROI pooling, allowing a single CNN passthrough on the entire image as opposed to separate CNN-based feature extraction in R-CNN. Faster-RCNN [**11**] further incorporated the region proposal step into the network itself, replacing selective search-based region proposal with a region proposal network (CNN). This added support for end-to-end training for both region proposal and bounding box prediction steps. Cascade R-CNN [**12**] improves detection performance by cascading multiple stages of classifiers, each focusing on different levels of difficulty. This approach enhances both precision and recall but may increase inference time. Mask R-CNN [**13**] further extends the Faster-RCNN model with an additional head for generating semantic segmentation maps for each RoI, facilitating the instance segmentation task.

1.3.2. Single-shot detectors

On the other hand, sliding-window-based approaches generally do away with the proposal generation stage and directly predict the bounding box, typically sacrificing accuracy for the inference speed. OverFeat [**14**] introduced the CNN version of the sliding window method, directly predicting bounding boxes after knowing the confidence scores of underlying categories of the top-most feature map. YOLO [**15**] introduced simultaneous class prediction and localization using the top-most feature map, where the predicted bounding boxes are shared between the multiple categories. SSD [**16**] further incorporated multi-scale prediction by having default boxes with different aspect ratios at each level of the feature map in the backbone. RetinaNet [**17**] introduced a feature pyramid network and focal loss to handle class imbalance, which led to better accuracy across all scales of object categories.

1.3.3. Anchor-free and Transformers

FCOS (fully convolutional one-stage object detection) [**18**] introduced a fully convolutional anchor box-free approach. Rather than relying on anchor boxes, for which the hyperparameters need to be tuned according to the dataset, this method relied on predicting the object centers rather than using anchor boxes. With the introduction of Transformer networks [**19**] with attention mechanism for understanding global context, DETR [**20**] introduced an encoder-decoder transformer-based architecture. It formulated the problem of object detection as direct set prediction, simplifying the process without the need for traditional approaches such as anchor boxes, non-max suppression, or region proposal networks. Without relying on these structures, DETR can adapt across various scales and aspect ratios.

1.4. Unknown-Aware Object Detection

Establishing robust object detectors capable of effectively identifying out-of-distribution (OOD) objects remains a relatively underexplored domain. Due to the inherent lack of supervision signals attributed to unknown data, leading models to exhibit overconfident predictions when confronted with OOD objects. Paper [**5**] introduces a novel framework named Spatial-Temporal Unknown Distillation (STUD), designed to imbue awareness of unknown entities. This framework operates by distilling unknown objects from videos collected in diverse, real-world settings, thereby instilling a valuable regularization effect on the model's decision boundary.

1.4.1. Unknown Distillation

Consider a pair of frames x_0, x_1 at timestamps t_0 and t_1. Let there be set of object features $\{h(x_0, b_i)\}_{i=1}^{N_0}$ and $\{h(x_1, b_j)\}_{j=1}^{N_1}$. The paper adopt a dissimilarity measurement using the L_2 distance between two features:

$$s_{i,j} = \frac{\|h^\wedge(x_0, b_i) - h^\wedge(x_1, b_j)\|^2}{2},$$

Where $h^\wedge(x_0, b_i)$ and $h^\wedge(x_1, b_j)$ are encoded feature vectors obtained by a small network using the object features $h(x, b)$ as input. The larger $s_{i,j}$ is, the more dissimilar the object features are. Lastly, we perform a weighted average of the object features from frame x_1, similar to attention:

$$\hat{o}_i = \sum_{j=1}^{N_1} \alpha_{i,j} h(x_1, b_j), \quad \alpha_{i,j} = \frac{e^{s_{i,j}}}{\sum_{k=1}^{N_1} e^{s_{i,k}}}.$$

Here, $\hat{o}_i$ represents the distilled unknown object in the feature space, corresponding to the i-th object at frame x_0. For Temporal aggregation, T frames are selected in the neighborhood of the key frame and distilled unknown is aggregated.

1.4.2. Unknown candidate selection

In the context of object detection network energy scoring, the determination of energy scores involves feeding the set of object features $\{h(x_1, b_j)\}_{j=1}^{N_1}$ into the prediction head, as follows:

$$E(x_1, b_j) = -\log \frac{1}{K} \sum_{k=1}^{K} \exp\left(f_k(h(x_1, b_j); w_{\text{pred}})\right),$$

where $f_k(h(x_1, b_j); w_{\text{pred}})$ represents the logit output of the k-way classification branch. A higher energy score signifies to greater out-of-distribution characteristics. Subsequently, a selection criterion is applied to identify objects falling within a specific percentile range $p\% \leq \text{Rank}(E(x_1, b_j))/N_1 \leq q\%$ among all objects. We apply this step individually per frame before doing temporal aggregation.

1.4.3. Uncertainty Regularization

The paper defines a regularization term based on the energy scores: denoted as $E(x, b)$, for the in-distribution (ID) objects, along with $E(\hat{o})$ for the distilled unknown object features. We train a classifier for ID vs unknown, with a weight coefficient θ_u. The GAN-like regularization loss, $L_{\text{uncertainty}}$, is calculated as follows:

$$L_{\text{uncertainty}} = \mathbb{E}_{\hat{o}\sim O}\left[-\log\frac{1}{1+\exp(-\theta_u \cdot E(\hat{o}))}\right] + \mathbb{E}_{(x,b)\sim D}\left[-\log\frac{\exp(-\theta_u \cdot E(x,b))}{1+\exp(-\theta_u \cdot E(x,b))}\right],$$

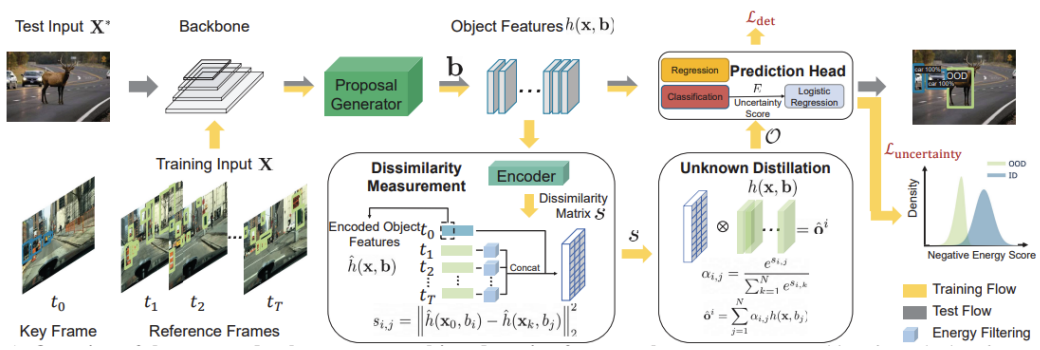


Fig. 1.2. The overview of STUD architecture[**5**]

Chapter 2

Dataset

As previously explained, we define our task evaluation under OOD detection framework. For this, we require

(1) In-Distribution dataset: Comprises of the object categories we are concerned about,

(2) Out-of-Distribution dataset: Comprises of object categories which do not overlap with in-distribution ones, used for evaluation

For the In-distribution dataset, we mainly use a custom dataset collected at E-smart, having 12 object categories typical of a self-driving dataset. For the out-of-distribution dataset, we mainly use the MS COCO dataset, consisting of 80 open-world object categories. We use the validation split from MS COCO.

We also use pre-trained Faster RCNN models using the following datasets as a pre-training checkpoint, before doing the respective fine-tuning:

(1) MS COCO dataset

(2) BigDetection dataset

The following sections cover the required details for each of these datasets.

2.1. MS COCO dataset [3]

MS COCO (Microsoft Common Objects in Context) is a large-scale image dataset containing 328,000 images of everyday objects and humans, each annotated with 80 object categories and 5 captions describing the scene. This provides a milestone for all literature researching large-scale open-world understanding. The dataset contains annotations you can use to train machine learning models to recognize, label, and describe objects. It encompases a plethora of tasks such as:

(1) Object detection: provides bounding boxes around objects of interests

(2) Object segmentation: Provides pixel-level annotations for objects of interests

(3) Instance segmentation: Detecting individual instances separately along with semantic segmentation
(4) Keypoint detection: Detecting specific points of interest such as structural points for objects
(5) Dense pose estimation: Estimating 3D poses of objects or people given the 2D images
(6) Panoptic segmentation: Semantic segmentation of all objects (includes objects in the background) combined with instance segmentation for objects of interest
(7) Image captioning: Each image has up to 5 caption descriptions describing the scene

The COCO dataset classes are divided into two main categories: "things" and "stuff."

"Things" classes include objects easily picked up or handled, such as animals, vehicles, and household items. "Stuff" classes include background or environmental items such as sky, water, and road.

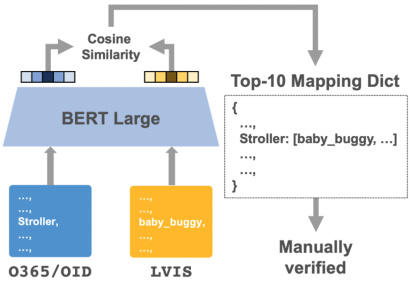
person	fire hydrant	elephant	skis	wine glass	broccoli	dining table	toaster
bicycle	stop sign	bear	snowboard	cup	carrot	toilet	sink
car	parking meter	zebra	sports ball	fork	hot dog	tv	refrigerator
motorcycle	bench	giraffe	kite	knife	pizza	laptop	book
airplane	bird	backpack	baseball bat	spoon	donut	mouse	clock
bus	cat	umbrella	baseball glove	bowl	cake	remote	vase
train	dog	handbag	skateboard	banana	chair	keyboard	scissors
truck	horse	tie	surfboard	apple	couch	cell phone	teddy bear
boat	sheep	suitcase	tennis racket	sandwich	potted plant	microwave	hair drier
traffic light	cow	frisbee	bottle	orange	bed	oven	toothbrush

Fig. 2.1. MS COCO dataset class list[**3**]

2.2. BigDetection dataset [4]

After the introduction of major datasets such as LVIS(1200+ categories), OpenImages(600 categories), and Objectt365(365), there were challenges to utilize all the datasets simultaneously. There existed overlapping categories with different semantic labels, having many fine-grained categories in some datasets, labeling or bounding box annotation errors, long-tailed distribution of annotation labels etc.

Many attempts were made to carefully design training strategies for multi-dataset object detection. BigDetection introduced a unified dataset utilizing all 3 datasets, carefully choosing the object categories as well as the annotation labels.



	Train		Val		Num. classes
	Num. images	Num. boxes	Num. images	Num. boxes	
LVIS [24]	100K	1.27M	19K	244K	1203
OpenImages [26]	1.74M	14.61M	41K	303K	600
Objects365 [44]	1.72M	22.89M	80K	1.06M	365
BigDetection	3.48M	35.96M	141K	1.58M	600

Fig. 2.2. (On left)The category mapping pipeline for label-space generation, demonstrating the merging process of semantic labels using BERT (On right) The statistics of individual datasets and after merging into BigDetection (Source: paper[**4**])

The paper proposed multiple stages for generating the unified label space:

(1) Class matching: Through the category mapping pipeline using the BERT language model embeddings and using cosine similarity. The paper first creates a top-10 map dictionary using LVIS(largest taxonomy) categories as a starting point.

(2) Class merging and de-duplication: These matched classes are merged and manually verified. Additionally, some categories representing the same semantical concepts but not captured in cosine similarity are also merged. The non-overlapping classes are added as additional categories. Some categories representing very fine-grained concepts are also merged into their parent categories. Finally, classes with too few samples are removed.

(3) Bounding box de-overlapping: The paper also looked into bounding box overlap to remove classes, based on IoU.

This merging led to final 600 classes, out of which 555 classes are frequent($>$ 100 samples). This solves the long-tailed distribution class imbalance problem.

2.3. E-smart custom dataset

In this study, we have a custom-curated dataset of video sequences of driving footage of cars. These were captured in various scenarios around Montreal and neighboring cities, including a diverse set of scenarios such as residential neighborhoods, urban and downtown areas, rural areas, and highways. To remove intrinsic biases introduced by the video capturing setup, different car makes and models were used and anonymized in the dataset. All the video sequences were captured by a single dash-cam mount deployed on the given car, which also captures variations.

From these captured video sequences, snapshots at manually chosen frames at various timestamps were extracted, to introduce diversity in captured scenes. These images vary in road conditions (rural, urban, highways), weather conditions (sunny, rain, snow), different regions using different standards and languages for road signs (French vs English) etc. We extract and annotate these images manually through a custom labeling tool for different object categories with their respective bounding boxes and additionally road and weather conditions as well.

The final Esmart dataset comprises of over 8000 images with 12 object categories, and a total of over 70000 bounding box instances. The detailed class distribution is given in Table 2.1.

S.No.	Class Category	Number of annotations
1.	bicycle	175
2.	bus	394
3.	car	42884
4.	lane	628
5.	lanes	7429
6.	motorcycle	166
7.	person	1618
8.	roadwork	7927
9.	speed limit	6887
10.	stop sign	1049
11.	traffic light	4437
12.	truck	4507
13.	Total	78101

Table 2.1. The class distribution of bounding boxes in E-Smart dataset

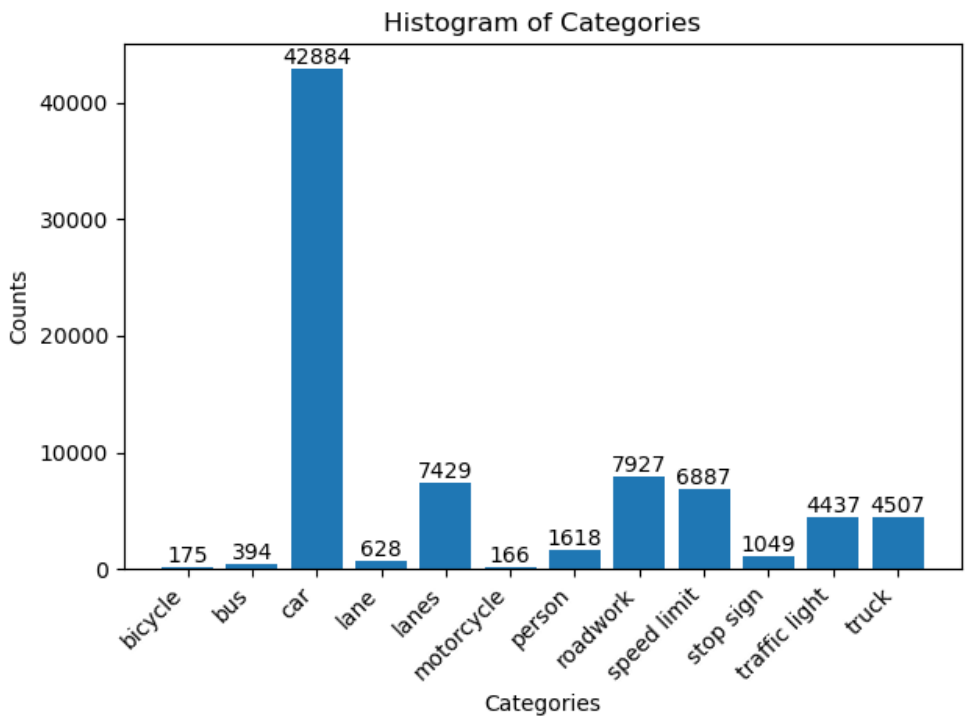


Fig. 2.3. Histogram of class-distribution in the E-Smart dataset.

2.4. Visualizing the object clusters: E-Smart dataset

We conduct an analysis to visualize the object clusters in latent representation space. We hypothesize that if the object clusters are sufficiently separated in representation space, we can use the said representation to learn in-distribution statistics and filter out OOD samples.

For this, we experiment with 2 different levels of representation of the Faster RCNN model. We experiment with the MS COCO-trained and BigDetection-trained models respectively, to extract the feature representation of objects.

2.4.1. Methodology

For the Faster RCNN model, we experiment with 2 levels of latent representation for a given bounding box, from the ROI head or box head block:

(1) Output of ROI pooling: This results into a (7 x 7) patch of 256 channels. We further flatten this into 12544-dimensional embedding.

(2) Output of pen-ultimate layer: This results into 1024 dimensional embedding for each of the bounding box.

We use t-SNE-based visualization for the above-mentioned representations for all the bounding boxes present in the train set, separated by the class labels. For the first 12544-dimensional feature embeddings, we first do dimensionality reduction using PCA, to 1024 dimensional representation. For the 1024-dimensional embeddings, we keep them as is and perform t-SNE.

The t-SNE representation gives a 2D visual representation of higher dimensional data, which can ve seen as the separability of object clusters of classes in the dataset. To obtain an objective metric for the separability of classes, we conduct a linear separability test. The idea here is the more classes are separable in higher dimensional representation space, the better the embeddings for the task of OOD detection, as these methodologies rely on cluster statistics for identifying the OOD samples.

We train an SVM with a linear kernel and measure the training accuracy as the metric for linear separability. The following sections cover these experiments performed for both the MS COCO pre-trained and BigDetection pre-trained models.

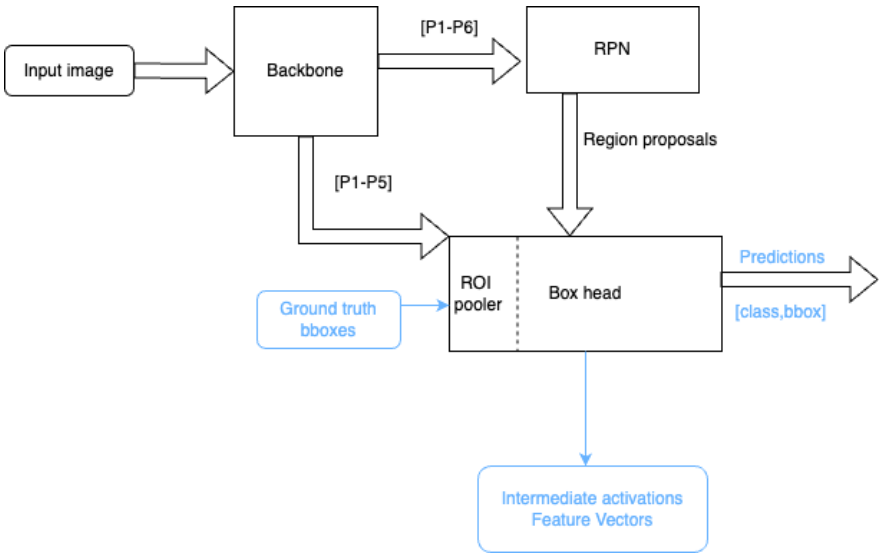


Fig. 2.4. The architectural diagram representing the algorithm to compute feature vectors for a given image and corresponding ground truth bounding boxes. Black signifies the traditional forward pass and blue signifies the modified pass-through. Note that the ROI pooler uses ground truth boxes instead of RPN proposals here.

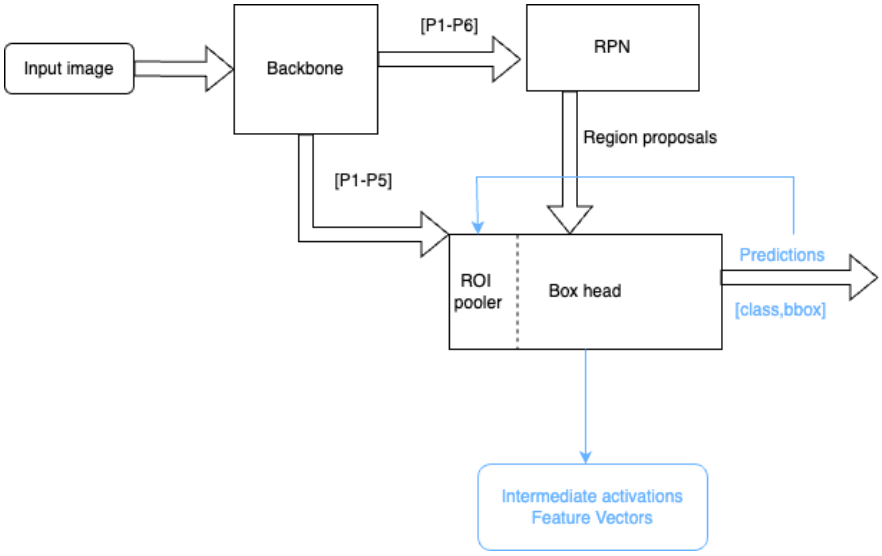


Fig. 2.5. The architectural diagram representing the algorithm to compute feature vectors for a given image and corresponding predicted bounding boxes. Black signifies the traditional forward pass and blue signifies the modified pass-through. Note that in the second pass, the ROI pooler uses ground truth boxes instead of RPN proposals to generate the feature vectors.

2.4.2. How to generate feature embeddings for a given bounding box?

For an image classification task, we classify the whole image belonging to a particular class. Hence, the feature embedding from an intermediate layer can be used for any class-related analysis and visualizations. This is simply not the case for object detection tasks, as there are multiple instances of objects present in the image.

To get the feature representation from an object detector, for a given bounding box, we utilize the ROI pooler of Faster-RCNN architecture. The ROI pooler is used to crop the feature representations generated by the backbone network, using the bounding box proposals from the Region Proposal Network. So, in the forward pass, we have 1000 proposals from the RPN, and any layer after ROI pooling will result in the embeddings for these proposals.

We utilize the ROI pooler differently. First, we store all the intermediate activations required for the ROI pooling layer in the forward pass. Based on the final predictions of the model, we reiterate the forward pass, but we pass these new predicted bounding boxes to the ROI pooler along with the original intermediate activations.

We can also choose to utilize the given ground truth bounding boxes for the task of understanding object clusters and separability, which is discussed in the following sections.

Algorithm 1 Computing the feature vector for given bounding box using Faster-RCNN

Require: Sample image x (optional ground truth bounding boxes B), layer index l in box head of Faster- RCNN

Do forward pass on sample x, store intermediate outputs of backbone (using hooks)

Pass the intermediate outputs of the backbone and bounding boxes (ground truth B **or** predicted $\hat{B}$) to box head (ROI pooling)

Do another pass-through box head and get intermediate activation at layer l(using hooks)

return List of feature vectors for test sample

$F(x) = [f_l(x,b) : \forall b \in B]$ **or** $[f_l(x,\hat{b}) : \forall \hat{b} \in \hat{B}]$

for ground truth or predicted bounding boxes, respectively.

2.4.3. MS COCO pretrained

As explained in the methodology section, we use a MS COCO pre-trained model, and fine-tune the box-head on our in-distribution dataset, E-Smart. Here, we present the testing of two different levels of feature extraction in the Faster-RCNN pipeline: the output of the ROI pooling layer, and the output of the penultimate layer.

To understand the latent representation embedding-space, we extract features from given ground truth bounding boxes and analyze them as listed in the methodology section.

2.4.3.1. ROI pooling layer (7x7x256 dim). Following Fig 2.6 shows the t-SNE plot after performing PCA to reduce dimensions to 1024. The feature vectors are color-coded with respective classes as given in the ground truth annotations. We note that the majority of the classes (7 out of 12) form their own clusters, albeit overlapping in the class boundaries. We also note that the classes without well-defined clusters belong to the minority class in the dataset: bicycle (175 samples) and motorcycle (166 samples).

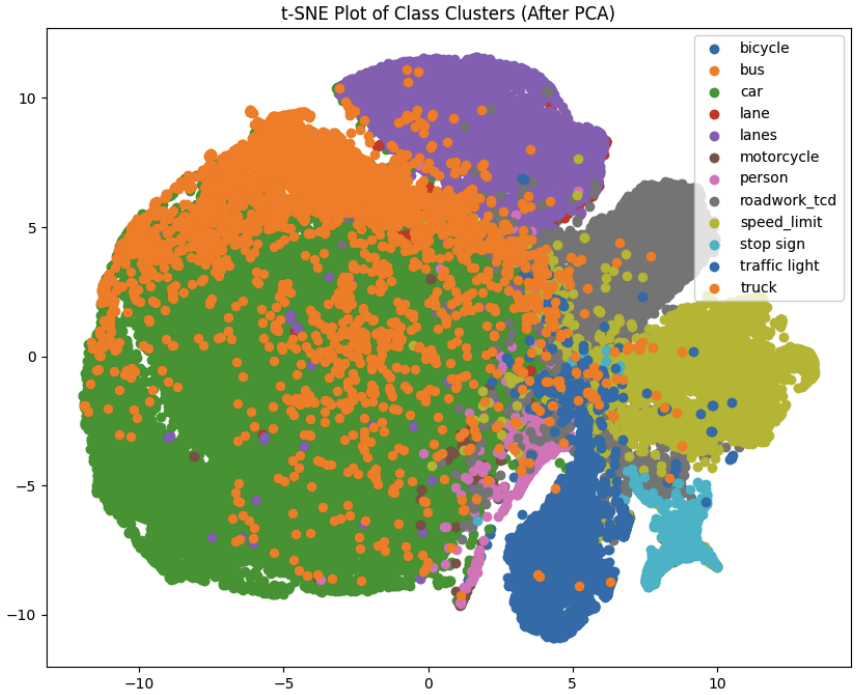


Fig. 2.6. The t-SNE visualization of feature vectors of ROI pooling layer after PCA dimensionality reduction to 1024 dims.

2.4.3.2. Penultimate layer (1024 dim). Following fig 2.9 shows the t-SNE plot of feature vectors extracted from the penultimate layer of COCO pre-trained Faster-RCNN. The feature vectors of 1024 dimensions are color-coded with respective classes as given in the ground truth annotations. We observe similar behavior as using ROI pooling layers, as discussed in the section above. The majority classes in class distribution such as car, bus, lanes, roadwork etc, again form well-defined clusters. Whereas minority classes have scattered points across the latent representation hyperspace.

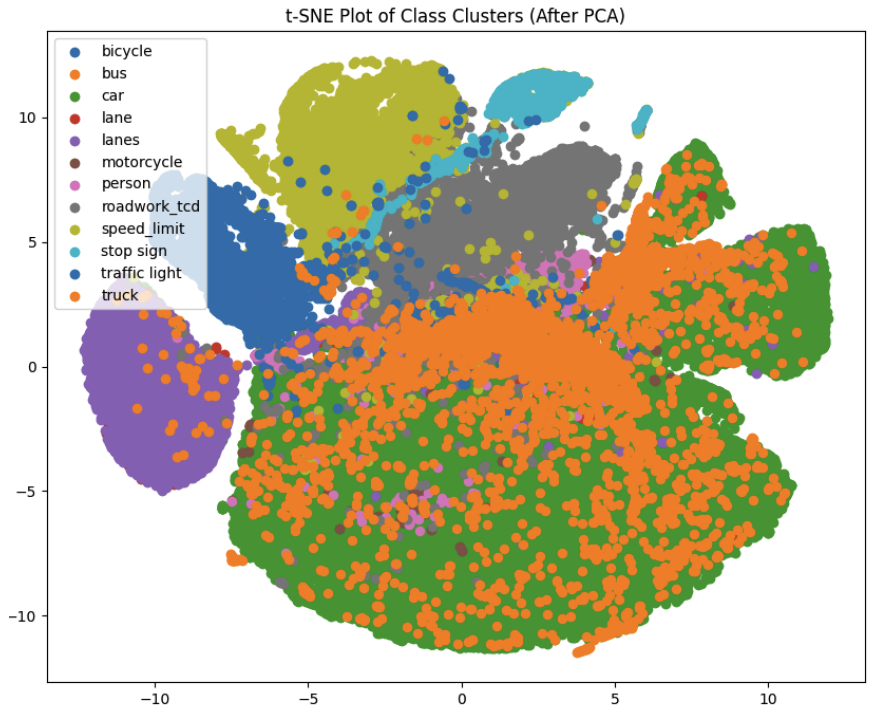


Fig. 2.7. The t-SNE visualization of feature vectors of penultimate layer, outputs are of 1024 dims.

2.4.3.3. Linear separability. To test the class separability, we perform the linear separability on the original extracted feature vectors. For comparative results with outputs of the penultimate layer, for ROI pooling layer generated feature vectors, we choose the embeddings generated after PCA reduces dimensions to 1024. We simply train a linear SVM on top of these feature vectors and report the training metrics, as demonstrating that the features are linearly separable in this hyper-space is sufficient. The results are listed in table 2.2.

Metric	**ROI pooling**	**Penultimate layer**
Accuracy	0.98	0.97
Precision	0.98	0.96
Recall	0.96	0.95
F1-score	0.98	0.97

Table 2.2. Training Metrics for MS COCO pre-trained model

We observe that all the reported training metrics are more or less similar for both ROI pooling layer and the penultimate layer. The important thing to note here the reported results for ROI pooling is after applying PCA. Utilizing these feature vectors in Mahalanobis distance-based detectors would require backpropagating the gradients, which would not be straightforward when PCA is involved.

2.4.4. BigDetection pretrained

We follow a similar process as the MS COCO section and use the BigDetection dataset as pre-training for the backbone and RPN. We finetune the ROI head with our in-distribution E-smart dataset. The following sections cover the results:

2.4.4.1. ROI pooling layer (7x7x256 dim). The feature vectors extracted after the ROI pooling layer, after doing dimensionality reduction using PCA form clusters in 1024-dimensional hyperspace. We observe that the classes form distinct clusters in t-SNE visualizations. Similar to COCO, we again observe that the majority classes form well-defined clusters.

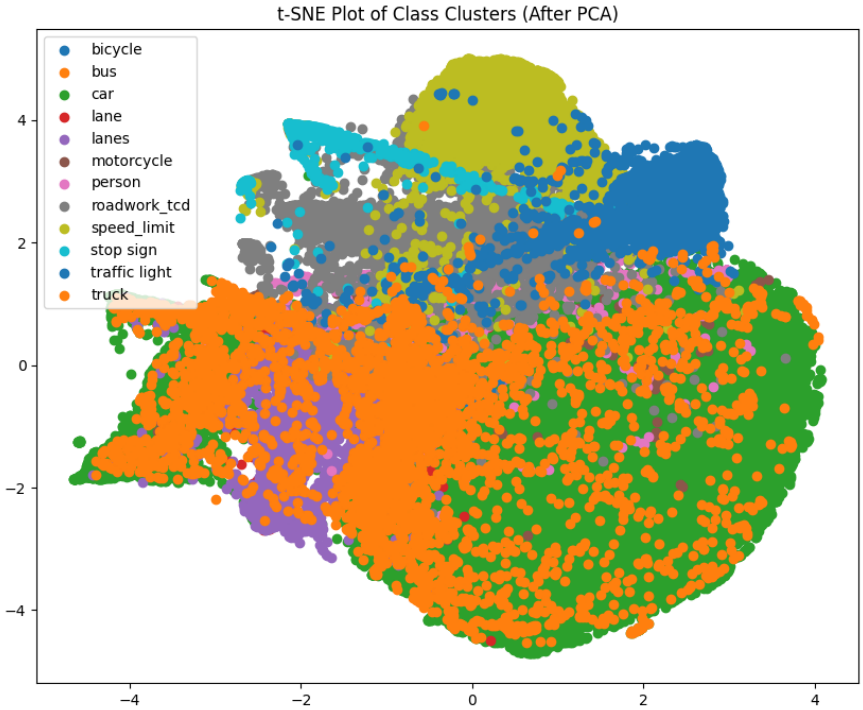


Fig. 2.8. The t-SNE visualization of feature vectors of ROI pooling layer after PCA, outputs are of 1024 dims.

2.4.4.2. Penultimate layer (1024 dim). Similar to ROI pooling layer, we observe the formation of well-defined clusters of respective classes for the ground truth bounding boxes in t-SNE visualization. The following figure supports this conclusion.

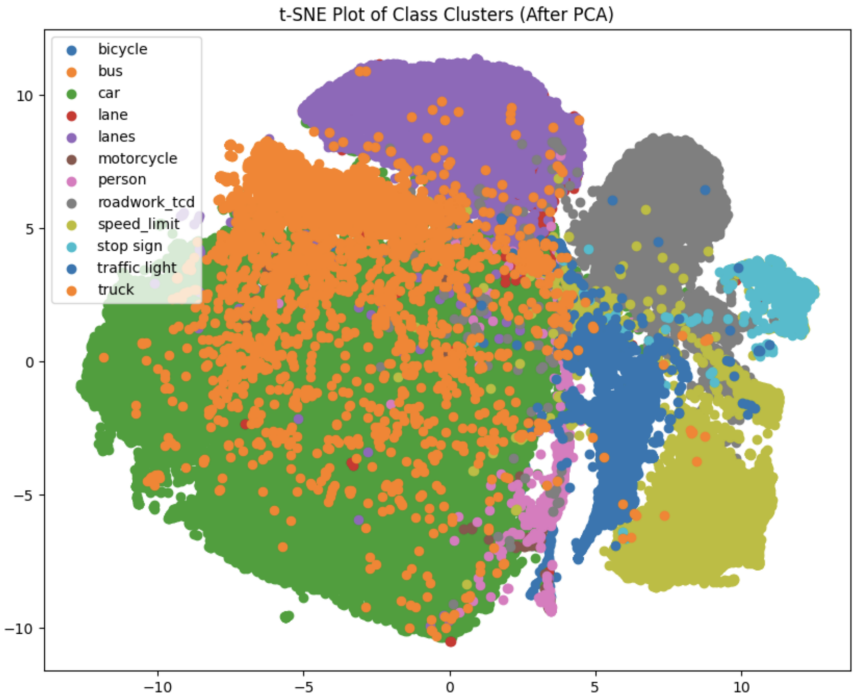


Fig. 2.9. The t-SNE visualization of feature vectors of ROI pooling layer after PCA, outputs are of 1024 dims.

2.4.4.3. Linear separability test. We also conduct the linear separability test with linear SVM as an objective measure of distinct cluster formation. The following table denotes the training metrics for this test, for the BigDetection pre-trained model.

Metric	**ROI pooling**	**Penultimate layer**
Accuracy	0.99	0.98
Precision	0.98	0.97
Recall	0.98	0.97
F1-score	0.98	0.97

Table 2.3. Training Metrics for BigDetection pre-trained model

2.4.5. Conclusion

From the t-SNE visualizations for hidden layer latent representation, we observe that the ground annotations bounding boxes form distinct clusters according to the class labels. Additionally, the linear separability test using linear SVMs further confirms the presence of well-defined boundaries in the latent hyper-space. This supports our hypothesis that both MS-COCO and BigDetection pretraining are effective in providing latent representation, for the task of OOD detection.

We further hypothesize that cluster-based statistical methods such as Mahalanobis distance can be used for in-distribution and out-of-distribution classification. We also observe that there is no significant difference in the resulting metrics of linear SVM for the two levels of latent representation. Hence, we utilize the penultimate layer for our further experiments, due to other reasons explained in section 2.4.3.3.

Chapter 3

Methodology and Model Pipeline

3.1. Understanding OOD samples in object detection

Let's take a look at OOD with the image classification task, the model with softmax classifier is forced to predict class probabilities from the given set of in-distribution class categories. Here, the predominant issue observed is that the model predicts wrong in-distribution class with a significant confidence score - known as the "Overconfidence issue". But with the task of object detection, we need to do simultaneous localization and class prediction. While training such localization models, as the models are not explicitly trained for localizing and detecting the OOD class categories, it is probable that OOD objects go undetected.

Such unknown objects going undetected can lead to unforeseen incidents in the context of autonomous driving systems. To avoid this scenario, it is pertinent that all such objects first get detected and the predicted probability scores indicate that the object is out-of-distribution. This process can be easily incorporated in two-stage detectors: Where the model first proposes regions of interest, which further leads to actual detection and localization. As opposed to single-shot detectors, where the detection and localization occur in a single step.

For the scope of our experiments, we take a look at the notable two-stage detector architecture: Faster-RCNN [**11**].

3.2. Faster RCNN model architecture

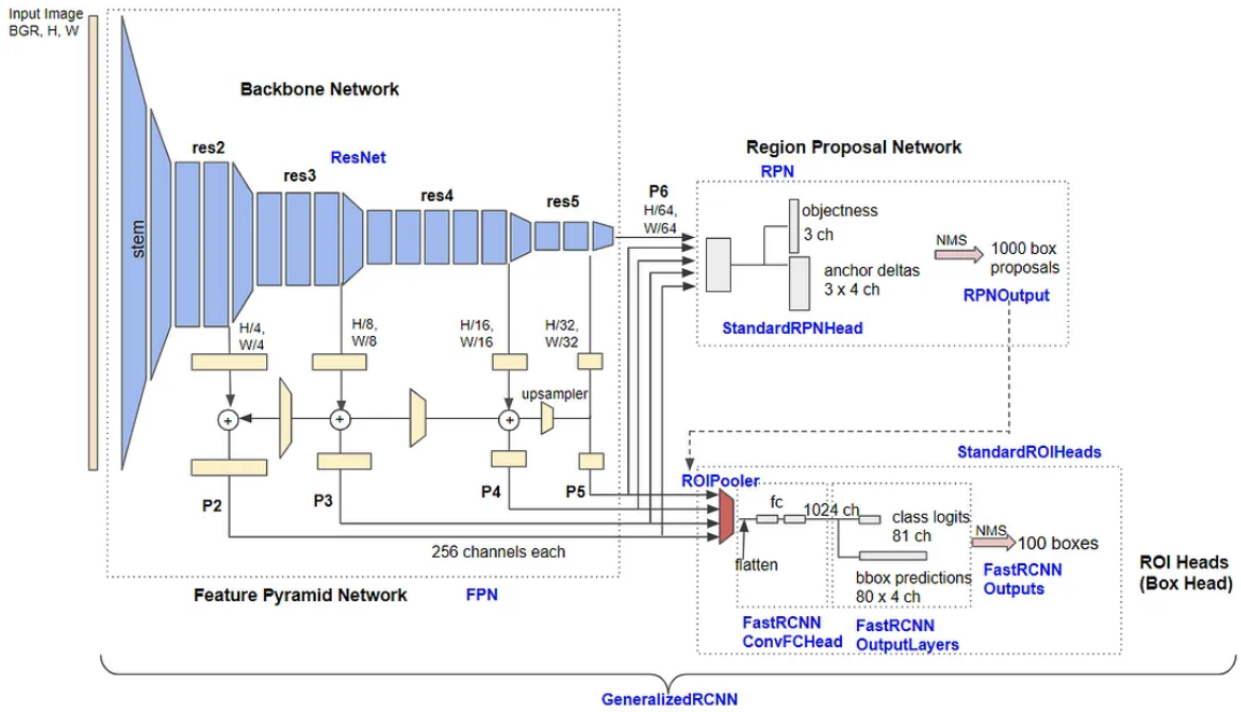


Fig. 3.1. Detailed architecture of Base-RCNN-FPN, blue signifies class names, source[**6**]

Faster-RCNN architecture is one of the most notable two-stage detectors for the localization (Bounding box detection) task. Here, we will take a look at the base-RCNN-FPN variant. It utilizes FPN(Feature Pyramid Network) to incorporate latent representation at various scales and gives it the capability to detect objects at multiple levels of resolution. It comprises of 3 blocks:

(1) Backbone network: extracts feature maps from the input image at different scales. Base-RCNN-FPN's output features are called P2 (1/4 scale), P3 (1/8), P4 (1/16), P5 (1/32) and P6 (1/64). Note that the non-FPN ('C4') architecture's output feature is only from the 1/16 scale.

(2) Region Proposal Network: detects object regions from the multi-scale features. 1000 box proposals (by default) with confidence scores are obtained.

(3) Box Head: crops and warps feature maps using proposal boxes into multiple fixed-size features, and obtains fine-tuned box locations and classification results via fully connected layers. Finally, 100 boxes (by default) in maximum are filtered out using non-maximum suppression (NMS).

3.2.1. Feature Pyramid Network

The feature pyramid network builds on top of the ResNet feature activations from multiple stages, having different resolutions. As the receptive field varies from narrow regions to wider regions down the stages, we effectively get feature representation at multiple scales.

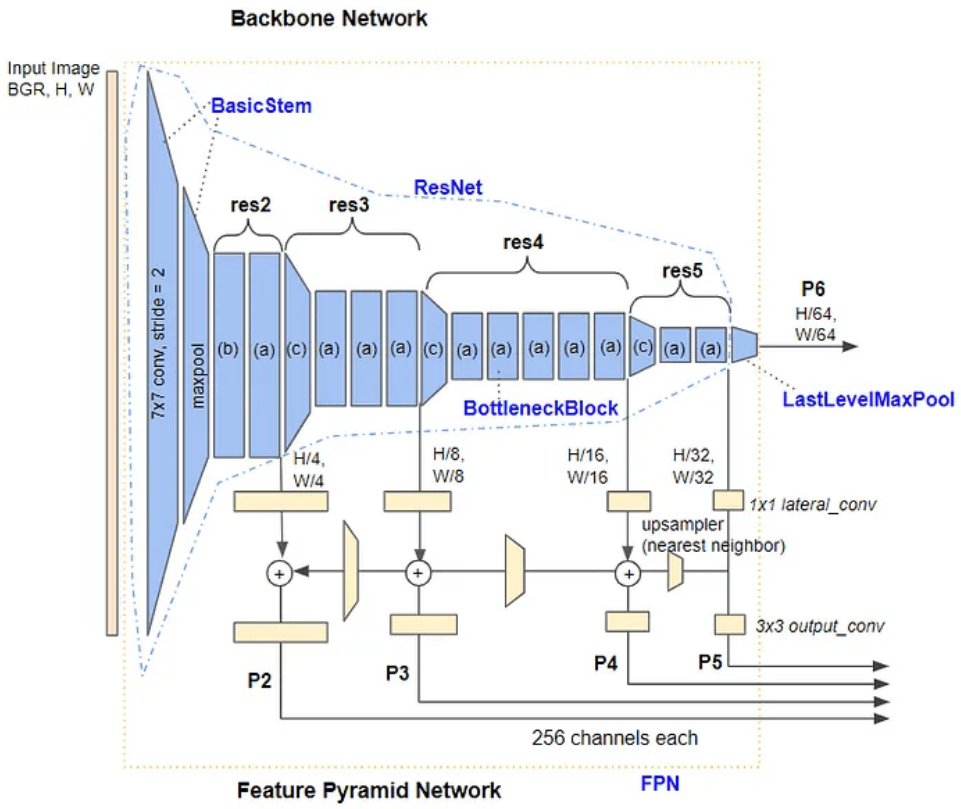


Fig. 3.2. Detailed architecture of Feature pyramid network with ResNet backbone. Blue labels are individual blocks in the architecture. (a),(b), (c) are different types of bottleneck blocks in ResNet.[**6**]

After getting intermediate activations at the end of each stage, we apply lateral convolution (1x1 convolution operation, with 256 channels). We also add the nearest neighbor upsampled activation from the previous stage to this. Finally, we perform the final output convolution (3x3 filter) and use these feature embeddings for further tasks, both in the region proposal network as well as for ROI pooling in the box head. This results into $\frac{1}{4}$ to $\frac{1}{32}$ scales of resolution of original image, in the P2 to P5 feature maps respectively.

One important distinction to note is in the final output of the ResNet backbone, namely P6 feature representation. We don't add activations from the previous stage and only do the max pooling to get $\frac{1}{64}$ downsampled feature map. We only use P6 for the region proposal network and not in the box head stage of the Faster-RCNN pipeline.

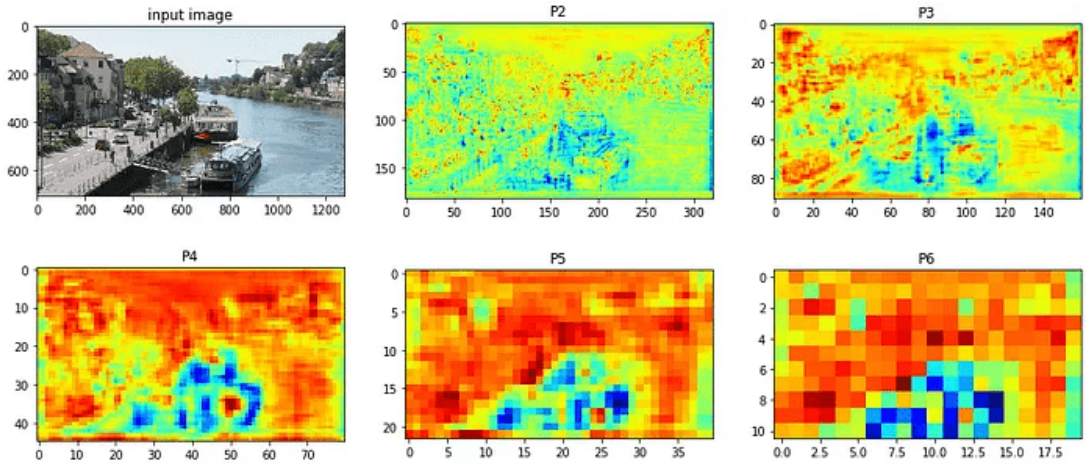


Fig. 3.3. Visualizing the feature map at each level from P2 to P6, for the given input image. These are 0th index channels.[**6**]

3.2.2. Region Proposal Network

The region proposal network adds a stage of proposing potential regions of interest for objects for the two-stage detection algorithm. The region proposal network is tasked with just proposing regions of interest, irrespective of what object category it might belong to. To train this network, we essentially train for foreground vs background classification and localization, with all the given object categories in the dataset treated as foreground class.

The objectness logit proposes the presence of an object at each grid point, for each of the predefined numbers of anchors. Based upon the IOU matcher, these anchor boxes are labeled into 3 categories for calculating regression on the anchor deltas:

(1) Foreground: The bounding box belongs to any of the object classes present in the dataset, with IOU with a ground truth box more than the predefined threshold (typically 0.7).

(2) Background: When the IOU is less than the predefined threshold (typically 0.3).

(3) Ignored: When IOU lies between these 2 thresholds, it can seen as any other grid point other than the central point of the object.

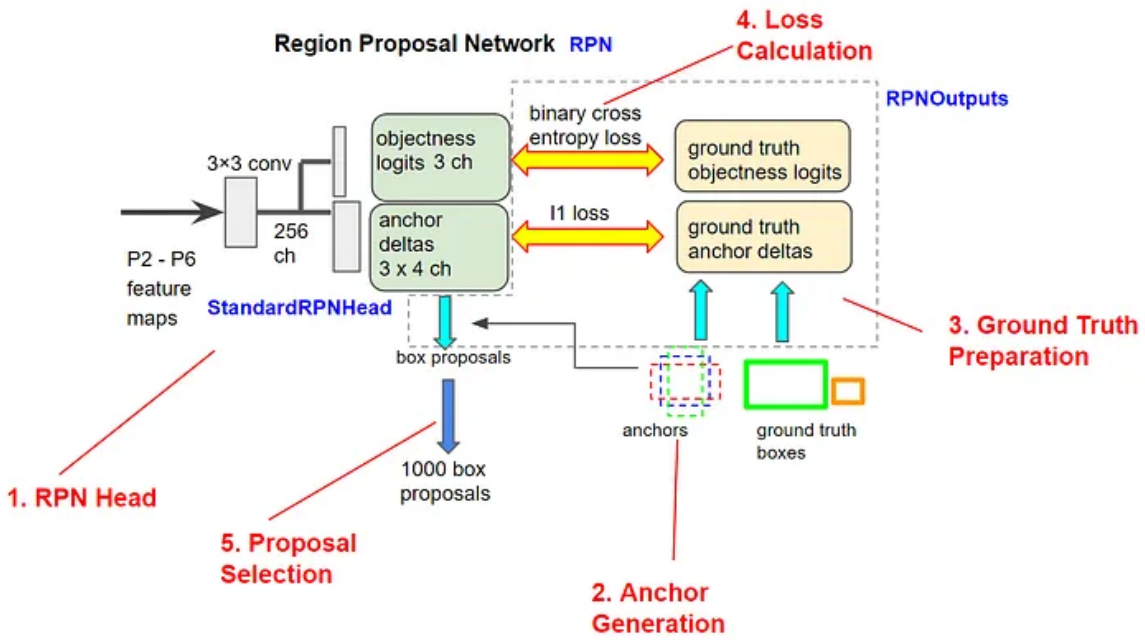


Fig. 3.4. Detailed architecture of Region Proposal Network.[**6**]

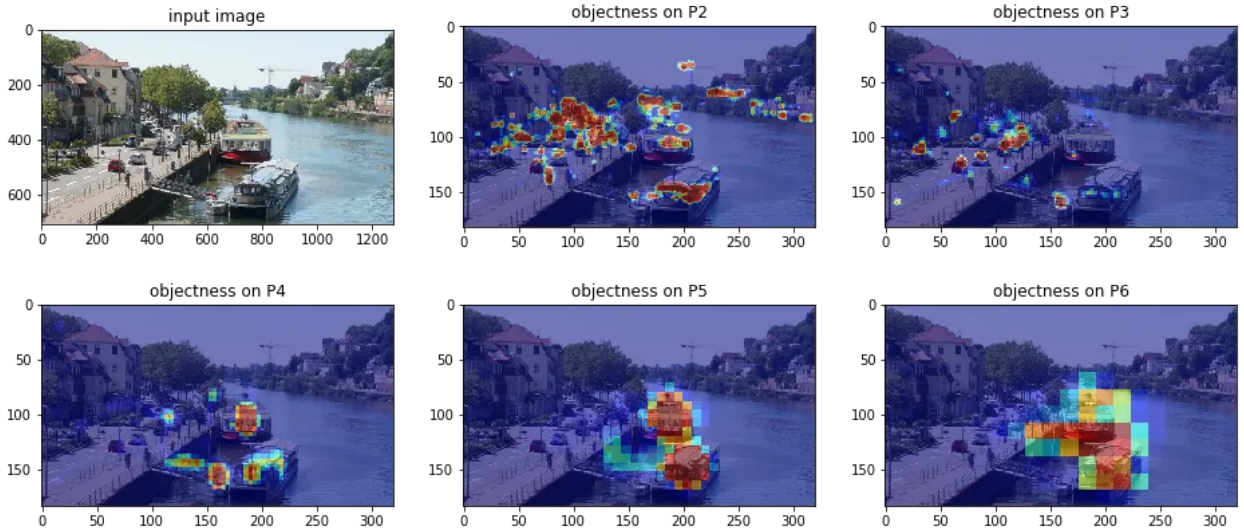


Fig. 3.5. Visualization of objectness map on the input image, for each feature map level. Sigmoid is applied to objectness logits.[**6**]

For training, two loss functions are used for the region proposal network:

(1) Objectness loss: this is binary cross-entropy loss applied to all points labeled as foreground (1) and background (0) above.

(2) Localization loss: this is regression loss (L1 loss) applied to only foreground grid points.

Lastly, before these region proposals are passed to the box head for final prediction, we perform Non-Max Suppression to prune the number of boxes to 1000.

3.2.3. Box Head

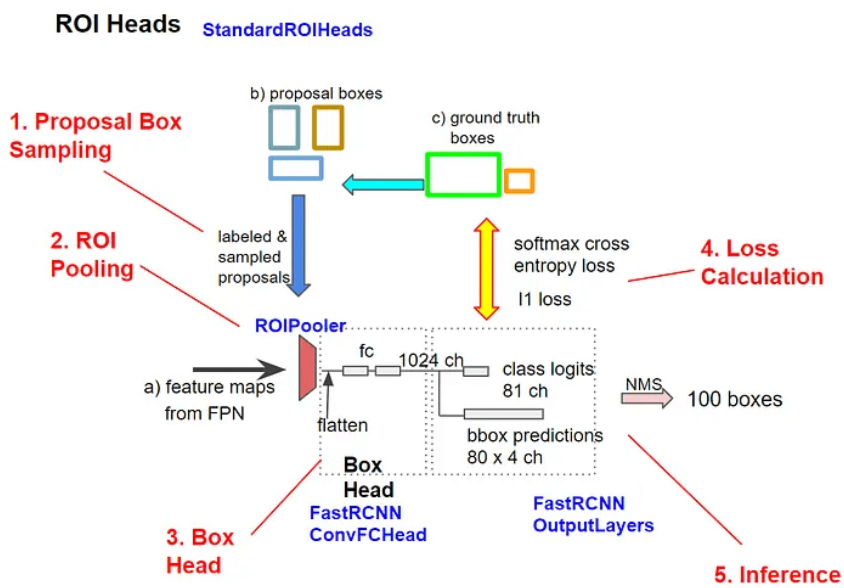


Fig. 3.6. Detailed architecture of Box Head.[**6**]

Box head takes the feature maps generated by the feature-pyramid network and all the proposed ROI bounding boxes from the region proposal network and provides final refined bounding box predictions with corresponding class labels.

First, we again do the ROI matching from all the ROI bounding boxes with ground truth bounding boxes, pruning again the potential number of boxes considered for further stage. the box head has an ROI pooling layer, which defines a heuristic to match the proposed ROI box with a corresponding feature map level provided by the FPN. Then it does the cropping, which results into a uniform shape feature map for all the bounding boxes. These feature maps are passed through few fully connected layers, and finally, we get class logits and bounding box predictions from the prediction head.

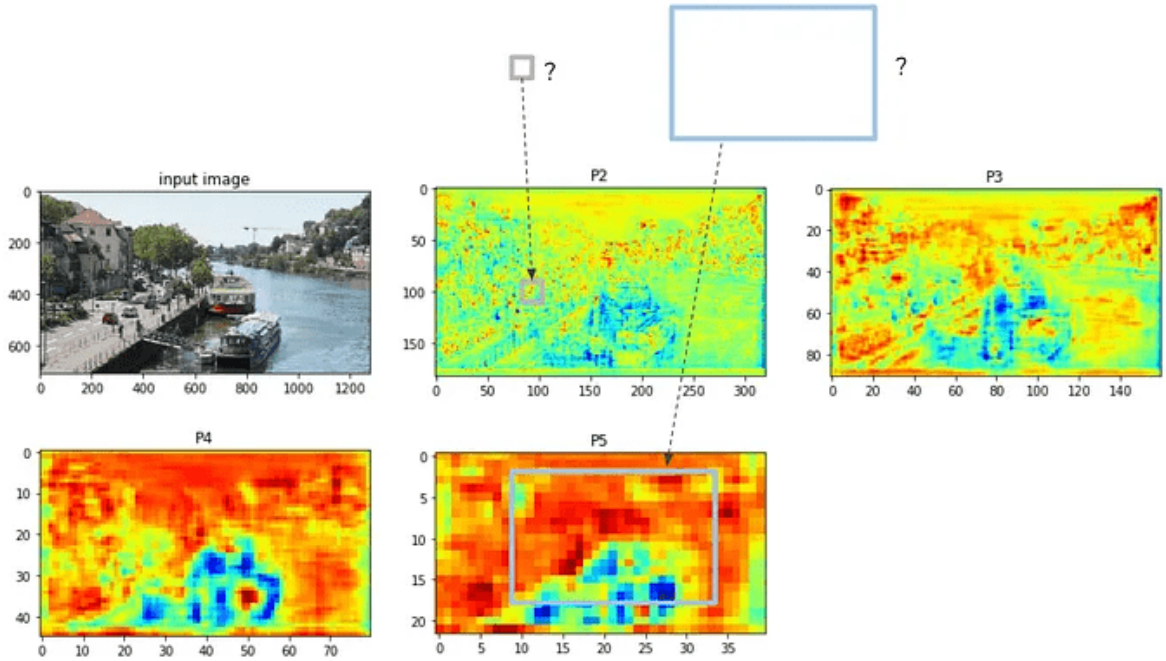


Fig. 3.7. Feature level assignment for ROI pooling. Here, the grey and blue boxes are assigned to levels P2 and P5 based on area heuristic, respectively.[**6**]

While training, we again utilize two loss functions:

(1) Classification loss: this is binary cross-entropy loss applied to all points labeled as foreground (1) and background (0) above. So this results in a (number of classes +1) total classes for multi-class classification.

(2) Localization loss: this is regression loss (L1 loss) applied to only foreground grid points.

During inference, we again apply Non-Max Suppression to all the predicted final bounding boxes.

3.3. Why do we need pre-trained RPN for OOD detection?

As discussed previously in section 4.1, we need to do simultaneous localization and classification for the object detection domain. To understand OOD samples in the object detection context, we first need to detect OOD objects in our pipeline, then we can utilize the OOD detection tricks for classification. Hence, having a separate Region Proposal network in the two-stage detectors is useful in these scenarios.

Suppose we train the RPN on an open-world object detection dataset, having a wide diversity of class labels. In that case, we can have a robust RPN detecting "Objectness" in any given sample, even the samples which are not explicitly seen in the In-Distribution set. For this reason, while training on the in-distribution set for object detection, we only finetuned the box head, leaving the weights of the feature pyramid network and region proposal network as is. For our experiments, we have utilized both MS COCO and BigDetection datasets as pre-training for the FPN and RPN of Faster-RCNN.

3.4. Fine-tuning on in-distribution dataset

As explained in the section 3.1, we utilize a two-stage detection process for OOD detection. We need the region proposal network to propose bounding boxes, even on unseen object categories w.r.t. our in-distribution dataset. To achieve this, we use large-scale open-world datasets as pre-training for backbone and region proposal networks and freeze the learned parameters. We only fine-tune the box head for final predictions on the in-distribution dataset. The following section covers the result of fine-tuning:

3.4.1. Results of fine-tuning on E-smart

We finetune the Faster-RCNN with FPN model, previously covered in section 3.2, using the train split of the E-smart dataset. we train for 9000 steps, with a batch size of 16. While keeping most of the hyper-parameters the same as the standard training script provided by Detectron2, with the exception of the learning rate scheduler. We use the same multi-step LR scheduler, with steps at 3600 and 4800, and warmup steps of 800.

For evaluation, we evaluate the held-out test set of the E-smart dataset. We use the MS COCO benchmark evaluation metric, mean average precision: taking the mean of average precision over all classes calculated using a precision-recall curve. This mAP is calculated

Metric	IOU Threshold	Area(Scale)	Max Detections	Value
Average Precision	0.5-0.95:0.05	all	100	0.402
Average Precision	0.50	all	100	0.571
Average Precision	0.75	all	100	0.449
Average Precision	0.5-0.95:0.05	small	100	0.276
Average Precision	0.5-0.95:0.05	medium	100	0.456
Average Precision	0.5-0.95:0.05	large	100	0.537
Average Recall	0.5-0.95:0.05	all	1	0.309
Average Recall	0.5-0.95:0.05	all	10	0.450
Average Recall	0.5-0.95:0.05	all	100	0.453
Average Recall	0.5-0.95:0.05	small	100	0.317
Average Recall	0.5-0.95:0.05	medium	100	0.509
Average Recall	0.5-0.95:0.05	large	100	0.585

Table 3.1. The results of COCO evaluation metric of finetuning on E-Smart dataset, using MS COCO pre-trained model. The highlighted rows depict the main metric as defined by the COCO challenge.

for a particular IOU threshold, so we again average over 10 threshold values lying between 0.5 to 0.95, incremented by 0.05. COCO also reports other different metrics such as AP and AR at IOU thresholds of 0.5, 0.75 as well as scale-wise AP and AR. The evaluation run for COCO and BigDetection pre-trained models is tabulated below:

As seen from the evaluation results from table 3.1 and table 3.2, the BigDetection pre-trained model performs better than the COCO pre-trained model. We see an increase of about $\sim 3\%$ on the average precision and recall (IoU 0.5-0.95:0.05), the COCO evaluation metric. We observe a similar increase in average precision (IoU 0.5), the PASCAL VOC evaluation metric.

Metric	IOU Threshold	Area(Scale)	Max Detections	Value
Average Precision	0.5-0.95:0.05	all	100	0.430
Average Precision	0.50	all	100	0.614
Average Precision	0.75	all	100	0.482
Average Precision	0.5-0.95:0.05	small	100	0.313
Average Precision	0.5-0.95:0.05	medium	100	0.481
Average Precision	0.5-0.95:0.05	large	100	0.553
Average Recall	0.5-0.95:0.05	all	1	0.333
Average Recall	0.5-0.95:0.05	all	10	0.484
Average Recall	0.5-0.95:0.05	all	100	0.487
Average Recall	0.5-0.95:0.05	small	100	0.358
Average Recall	0.5-0.95:0.05	medium	100	0.539
Average Recall	0.5-0.95:0.05	large	100	0.605

Table 3.2. The results of COCO evaluation metric of finetuning on E-Smart dataset, using BigDetection pre-trained model. The highlighted rows depict the main metric as defined by the COCO challenge.

3.4.1.1. Visualizing the results on OOD samples. To understand the results of these pre-trained models, we also visualize them by running inferences on randomly scrapped images from the internet. We use keywords such as "road under construction Montreal", "animals crossing the road", "bad road and weather condition" etc. to gather scenarios we are most concerned about for autonomous driving settings in Canada and USA.

Fig. 3.8. a) Elephant crossing road b) MS COCO output c) BigDetection output

Fig. 3.9. a) Yak crossing road b) MS COCO output c) BigDetection output

Fig. 3.10. a) Deer crossing road b) MS COCO output c) Big Detection output

Fig. 3.11. a) Bad visibility on road b) MS COCO output c) BigDetection output

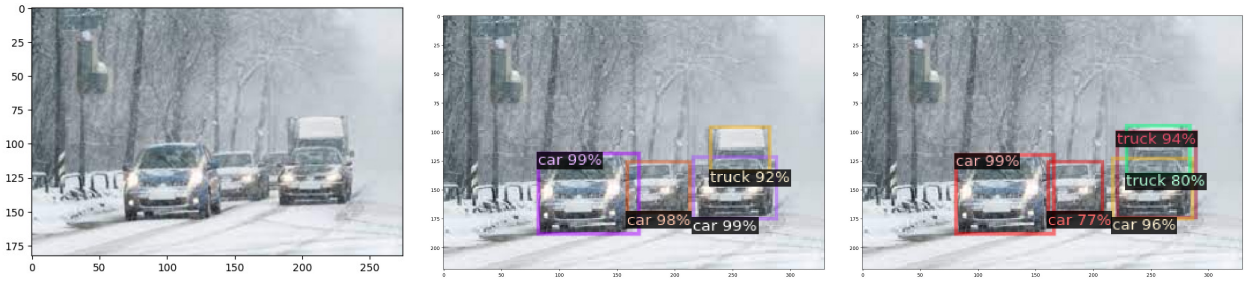


Fig. 3.12. a) Cars on snowy road b) MS COCO output c) BigDetection output

Fig. 3.13. a) Road under construction b) MS COCO output c) BigDetection output

Fig. 3.14. Visualizing the results of the fine-tuned models on images containing OOD samples

We make the following observations based on the outputs of the pre-trained models:

(1) Both the models miss out on identifying the animals across all the scenes, as it's an OOD object category for them.

(2) Overall, BigDetection pre-training leads to better results for small objects. It identified small object (stop-sign) in 'elephant crossing road' as well as 'bad visibility on road' pictures.

(3) BigDetection pre-trained model also made fewer errors on 'road under construction'. Both models misidentify a car at same location, but the COCO-pre-trained model does it for 2 additional locations. It also misses a clearly visible truck, which the BigDetection pre-trained model correctly identifies.

Overall, our hypothesis is validated about the challenges with OOD identification in object detection:

(1) The models don't identify the OOD samples present, so we need to have additional measures to identify them first.

(2) Secondly, there is a miss-classification of OOD samples as an in-distribution sample, same as the OOD detection in classification literature.

3.5. OOD detection with Mahalanobis distance

We first look into the methodology proposed in [**1**], which utilizes Mahalanobis distance-based OOD detection for the classification domain. This method does not need any additional retraining of the model, as covered in section 1.2.5. We adapt this method for object detection literature with some modifications, which is covered in the next section.

The paper proposes to utilize the Mahalanobis distance in the following way:

(1) Select a latent representation (or combination of them) from an intermediate layer of the model.

(2) For the given in-distribution train set, first calculate per-class means of embedding of given training samples with class labels. Calculate the covariance (the inverse of covariance to be precise) matrix over all the training sample embeddings, utilizing the means calculated above.

(3) For a given test sample, we first compute its Mahalanobis distance to all the class clusters using the class means and covariance. The closest class belongs to the minimum Mahalanobis distance.

(4) Then we compute its gradient w.r.t. to the input image by backpropagating the Mahalanobis distance of the closest class. We add a scalar multiple of this noise gradient to input and calculate the Mahalanobis distances again, where the maximum value corresponds to the confidence score.

Based on the confidence score computed above, a simple thresholding can be used for OOD detection.

Algorithm 2 Computing the Mahalanobis distance-based confidence score

Require: Test sample x, weights of logistic regression detector α_l, noise ϵ, and parameters of Gaussian distributions $\{\hat{\mu}_{l,c}, \hat{\Sigma} : \forall l, c\}$

Initialize score vectors: $M(x) = [M_l : \forall l]$

for each layer $l \in 1, \ldots, L$ **do**

Find the closest class: $\hat{c} = \arg\min_c \left(f_l(x) - \hat{\mu}_{l,c}\right)^T \hat{\Sigma}^{-1} \left(f_l(x) - \hat{\mu}_{l,c}\right)$

Add small noise to test sample: $\hat{x} = x - \epsilon \text{sign}\left(\nabla_x \left(f_l(x) - \hat{\mu}_{l,\hat{c}}\right)^T \hat{\Sigma}^{-1} \left(f_l(x) - \hat{\mu}_{l,\hat{c}}\right)\right)$

Computing confidence score: $M_l = \max_c \left(-\left(f_l(\hat{x}) - \hat{\mu}_{l,c}\right)^T \hat{\Sigma}^{-1} \left(f_l(\hat{x}) - \hat{\mu}_{l,c}\right)\right)$

end for

return Confidence score for test sample $\sum \alpha_l M_l$

3.6. Adapting Mahalanobis distance for the Faster-RCNN

For adapting the above-explained Mahalanobis distance-based OOD detection to object detectors, the core issue is that we have multiple instances of bounding boxes present in each image, which might belong to different classes. We can consider this individual bounding box prediction as one image, then the methodology can be applied.

To implement this idea, we utilize the ROI pooling layer present in the Faster-RCNN architecture. We utilize the same strategy to generate the feature embeddings as mentioned in the data visualization section 3.2.2, for bounding boxes proposed by the RPN. Following is the step-by-step process to get Mahalanobis distance:

(1) First, we generate the feature embeddings for all the bounding boxes present in the train set. We compute the class means and the covariance matrix.

(2) For a given image, first we get all the bounding boxes as generated by RPN. We get the feature representation for them. Additionally, we freeze the network with the computational graph of the forward pass.

(3) We calculate the closest class using the minimum Mahalanobis distance, and back-propagate it through the frozen computational graph and get the gradients w.r.t. the input image, for each of the proposed bounding boxes by the RPN.

(4) We initialize another Faster-RCNN model by copying the weights. This model is used for the second forward pass of the modified image, computed by adding a scalar multiple of the gradient. We calculate the confidence score with the modified image, using the new feature embedding generated by the same proposed bounding box. We do this process iteratively for each proposed bounding box.

The following pseudo-code defines the algorithm for computing the Mahalanobis distance-based confidence score using a particular hidden layer in Faster-RCNN:

Algorithm 3 Computing the Mahalanobis distance-based confidence score for Faster-RCNN

Require: Test sample x, noise ϵ, layer index l in box head, and pre-computed parameters of Gaussian distributions $\{\hat{\mu}_c, \hat{\Sigma} : \forall c\}$

Initialize a copy Faster-RCNN $\hat{f}(x)$ with same weights as the pre-trained model $f(x)$

Compute proposed bounding boxes $\hat{B}$ on sample x, and corresponding feature vectors at layer l using algorithm 1, $[f_l(x,\hat{b}) : \forall \hat{b} \in \hat{B}]$

Freeze the computational graph of $f(x)$ for multiple back-propagations

for each proposed box $\hat{b} \in 1, \ldots, \hat{B}$ **do**

Find the closest class: $\hat{c} = \arg\min_c \left(f_l(x,\hat{b}) - \hat{\mu}_c\right)^T \hat{\Sigma}^{-1} \left(f_l(x,\hat{b}) - \hat{\mu}_c\right)$

Add small noise to test sample: $\hat{x} = x - \epsilon \text{sign}\left(\nabla_x \left(f_l(x,\hat{b}) - \hat{\mu}_{\hat{c}}\right)^T \hat{\Sigma}^{-1} \left(f_l(x,\hat{b}) - \hat{\mu}_{\hat{c}}\right)\right)$

Computing confidence score using copy Faster-RCNN $\hat{f}_l$:

$M_{\hat{b}} = \max_c \left(-\left(\hat{f}_l(\hat{x},\hat{b}) - \hat{\mu}_c\right)^T \hat{\Sigma}^{-1} \left(\hat{f}_{l,\hat{b}}(\hat{x}) - \hat{\mu}_c\right)\right)$

end for

return Confidence scores for test sample x, $M(x) = [M_{\hat{b}} : \forall \hat{b} \in \hat{B}]$

The following flow diagrams denote the algorithm in 2 stages: computing gradient w.r.t. to input image and final Mahalanobis distance-based confidence score calculation. This computation is repeated per the proposed bounding box and the computation graph is frozen and re-utilized for all gradient computations.

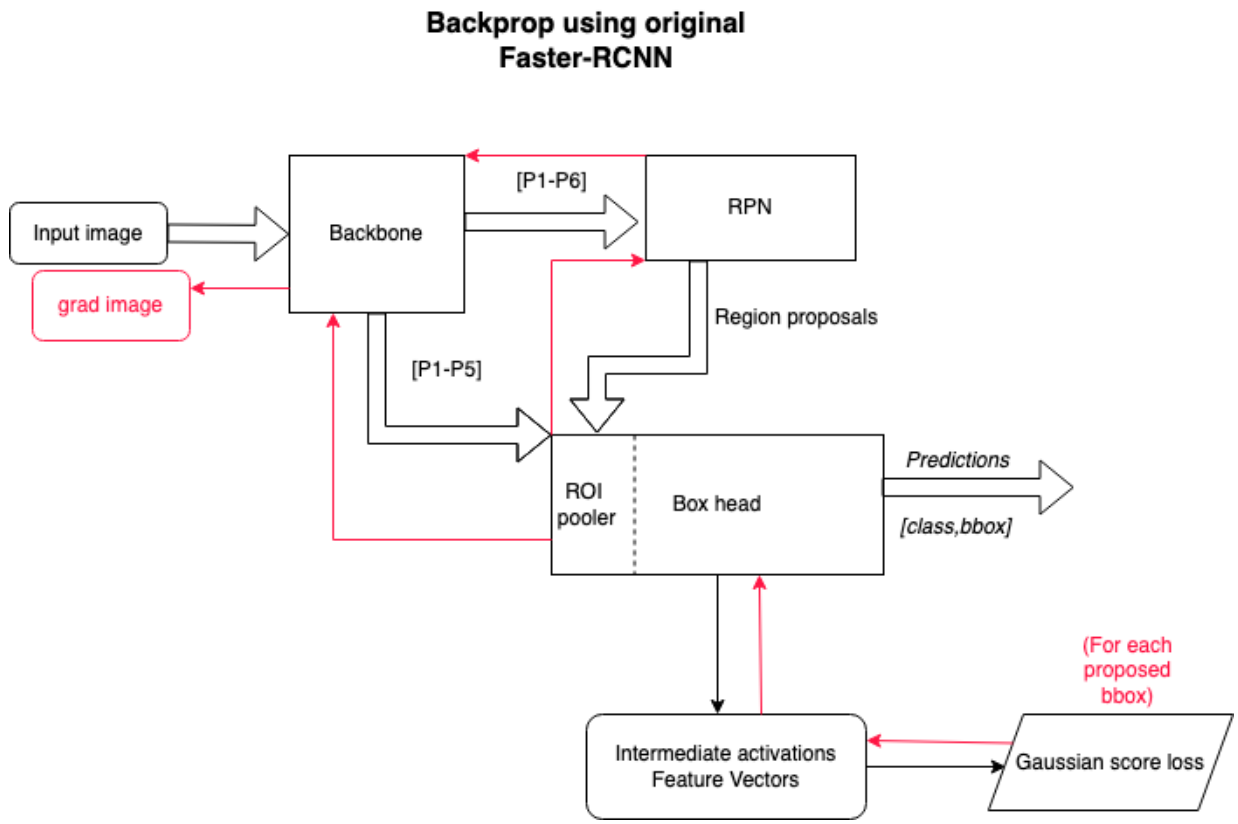


Fig. 3.15. Schematic for calculating the gradient image w.r.t. input using Mahalanobis distance loss per proposed bounding box. Black signifies the traditional forward pass, red signifies the backdrop computational flow.

3.6.1. Using pre-trained model predictions instead of region proposals

While experimenting with region proposals we found:

(1) The proposal boxes need finetuning. Sometimes multiple proposal boxes are overlapping, or they cover much broader areas than objects of interest.

(2) Due to some object categories being OOD concerning the pre-training dataset, we don't get proposals for such ID objects, even though the finetuned box-head detects them.

To solve these issues, we instead propose to use both the predictions of the pre-trained as well as fine-tuned models for OOD proposals. We then calculate the Mahalanobis distance-based confidence score for classifying the said proposals. Using predictions from both pre-trained

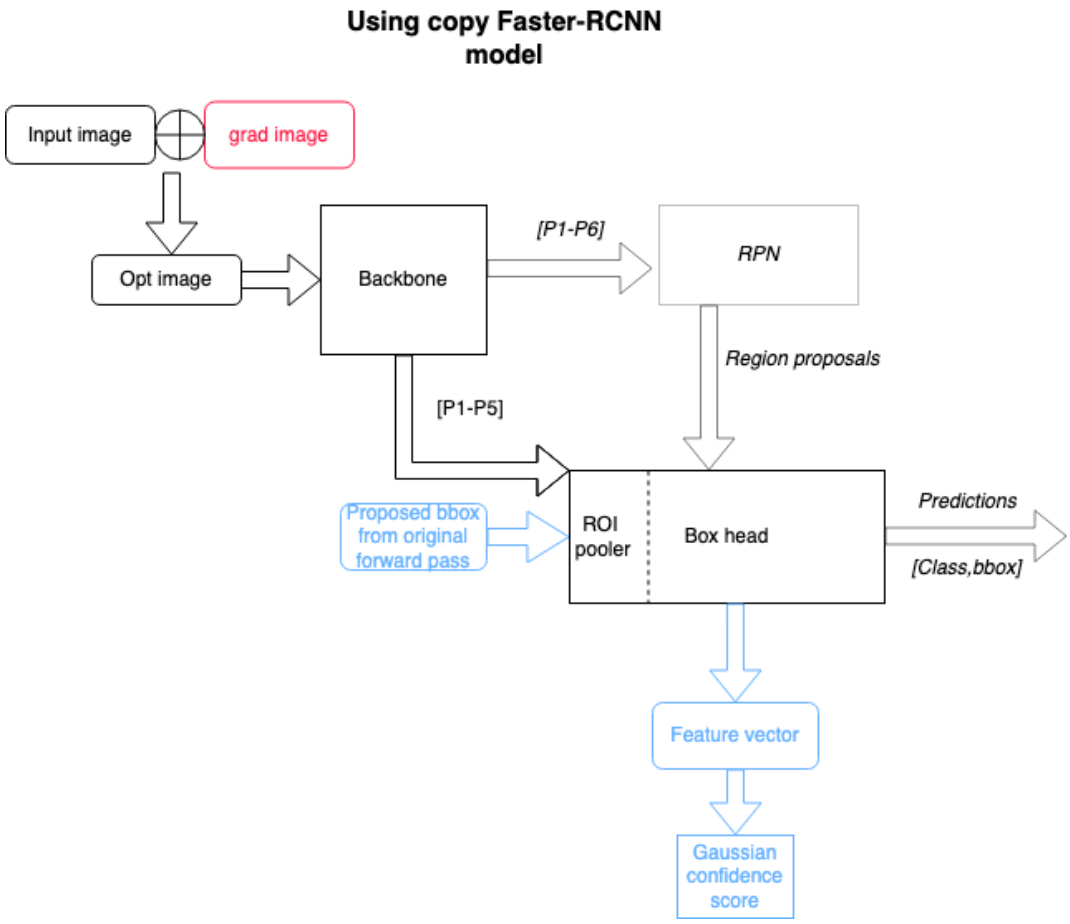


Fig. 3.16. Schematic for computing the final confidence score based on Mahalanobis distance, using copy model and given bounding box. Black signifies the traditional forward pass, blue denotes the second pass-through for calculating the Gaussian score. Note that RPN and box head from the traditional forward pass are blurred (and are italics) as they are not used.

box-head and finetuned box-head bounding box overlap for same objects. We employ non-max suppression, using OOD confidence predictions for filtering. Figure 3.17 shows the new algorithm for gradient computation, and for calculating the final confidence scores we use the same methodology as the previous section, explained in figure 3.16.

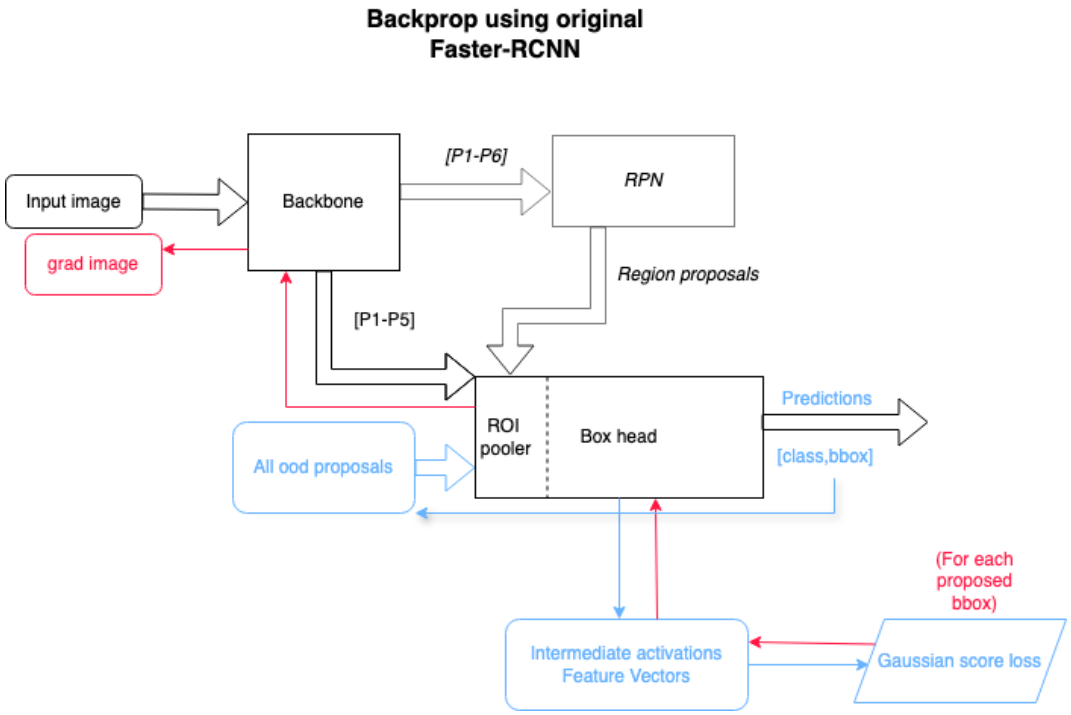


Fig. 3.17. Schematic for computing the final confidence score based on Mahalanobis distance, using copy model and given bounding box. Black signifies the traditional forward pass, blue denotes the second pass-through for calculating the Gaussian score. Red denotes the gradient flow to the input image. Note that RPN from the traditional forward pass is blurred (and in italics) as they are not used in gradient computation.

The following section will cover the experimental results of this proposed methodology. We also visualize our results using the images covered in section 3.4.1.1.

3.7. Results

In the following section, we demonstrate the results of our earlier methodology with the region proposals and issues that arise with it. We also discuss the improved method of using the pre-trained model for proposals and then the finetuned model for OOD detection in the later section.

3.7.1. Issues with just using region proposals

Let's look at the sample output of RPN. We typically train the RPN to generate top 1000 region proposals, which are then refined by the box-head for the final predictions. For our OOD detection methodology, we only apply the algorithm to the top 100 region proposals, which we filter out further using Non-max Suppression to the final top 20 proposals. We also show the output of our OOD detection pipeline.

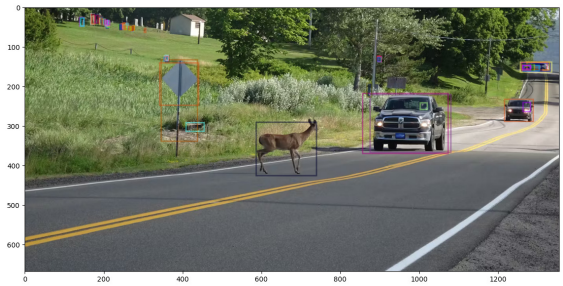


Fig. 3.18. (a)Output of region proposals

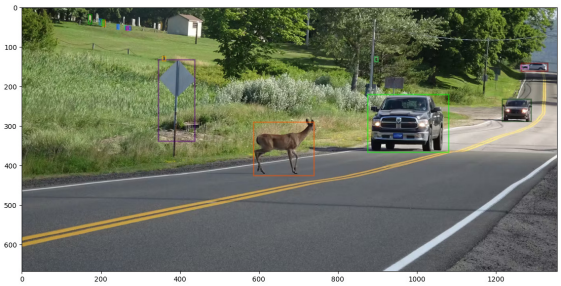


Fig. 3.19. (b)Regions after NMS

Fig. 3.20. (c)Output of OOD detection

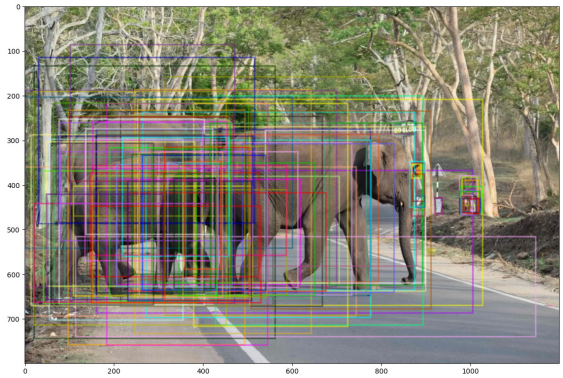


Fig. 3.21. (a)Output of region proposals

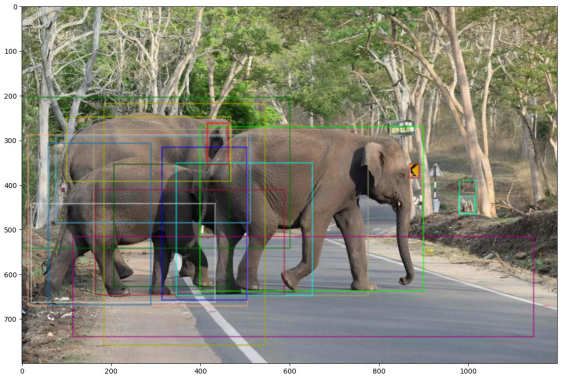


Fig. 3.22. (b)Regions after NMS

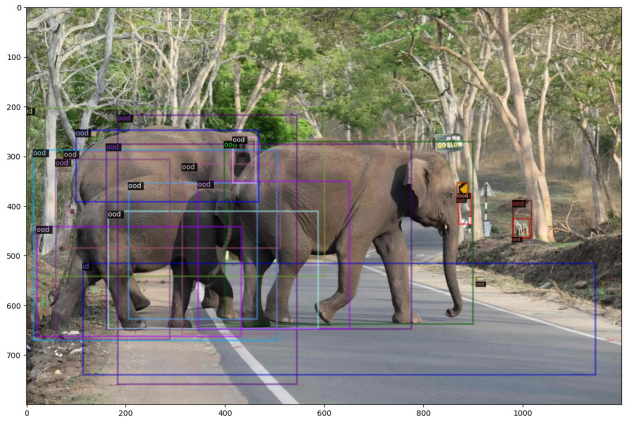


Fig. 3.23. (c)Output of OOD detection

From the above figure, we can see that the proposals from RPN are not always good, they need further finetuning. Even after NMS, there are multiple bounding boxes that are too broad or cover the object partially, there are even numerous bounding boxes for the same object. In the following section, we show that using predictions from both the pre-trained and finetuned models, we have much better proposals that also propose OOD object regions. We show that we can tackle both problem objectives we have defined earlier.

3.7.2. Visualising the OOD Detection

To better understand the proposed algorithm, we demonstrate its capability for OOD detection on a few sample images we're most concerned about for identifying unknown objects. These are the same images we have discussed in section 3.4.1.1. As discussed, these images cover two important scenarios:

(1) OOD objects getting miss-classified as in-distribution class objects.

(2) OOD objects are not getting identified.

Let's analyze the sample images and outputs of OOD detection, which demonstrate how we're able to solve both of the issues with the proposed algorithm.

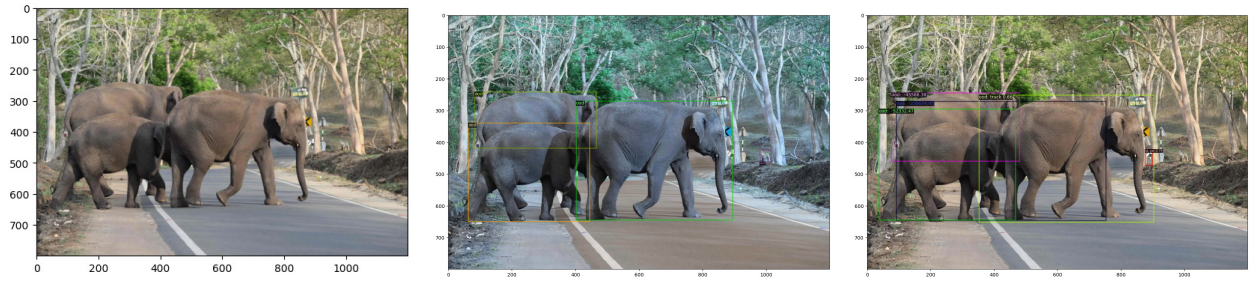


Fig. 3.24. a) Image before OOD detection b) MS COCO output c) BigDetection output

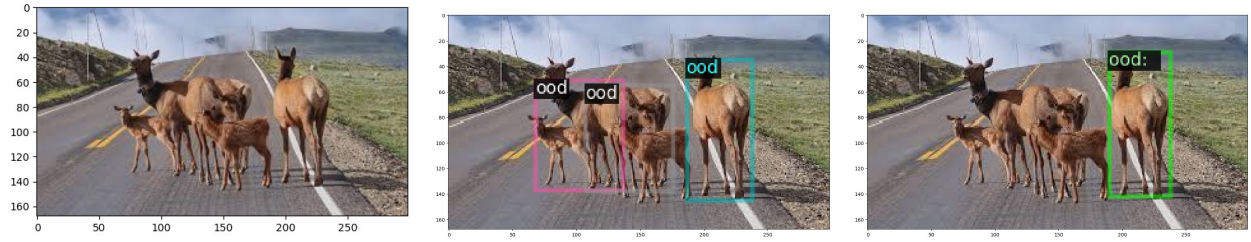


Fig. 3.25. a) Image before OOD detection b) MS COCO output c) BigDetection output

From the above images we can infer that using the prediction boxes of pre-trained models helps add the previously undetected objects from the box-head, which was finetuned on the ID dataset.

This sample image, 'Yak crossing road', consists of both ID and OOD objects. From the finetuned models' outputs, we can see that the 'Yak' wasn't identified before. Now, its identifying the OOD objects as well as distinguishing between OOD and ID classes.

Fig. 3.26. a) Image before OOD detection b) MS COCO output c) BigDetection output

Fig. 3.27. a) Image before OOD detection b) MS COCO output c) BigDetection output

Fig. 3.28. a) Image before OOD detection b) MS COCO output c) BigDetection output

In this sample, 'Road under construction', although some of the construction signs and cones are identified correctly by the finetuned model, it also falsely detects 'car' for what seems to be construction debris. The OOD detection correctly identifies and labels those detected objects as "OOD". We also have the detected class and the respective confidence scores annotated with the bounding boxes.

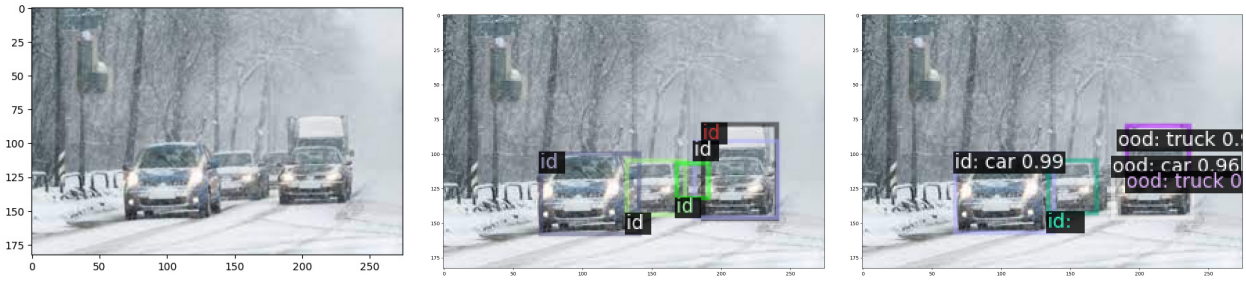


Fig. 3.29. a) Image before OOD detection b) MS COCO output c) BigDetection output

Fig. 3.30. Qualitative assessment of pre-trained models with OOD detection

We also observe that the outputs of OOD detection using the COCO pre-trained model yield overall better results than the BigDetection model. While experimenting with different confidence score thresholds we observe that the Mahalanobis distance-based confidence score does not necessarily produce clear separation in the BigDetection model. Whereas, it performs very well for the COCO pre-trained model.

We believe this could be due to the nature of the dataset itself, which would be an interesting research subject for further exploration.

Chapter 4

Conclusion

4.1. Discussion

For autonomous or driver-assistant systems, it is pertinent for safety reasons that the system should warn the users when it's uncertain. Traditional object detection only deals with improving its detection capabilities, based on a pre-defined set of object classes. It's important that the algorithm also detects and classifies unknown objects. In this project, we tackle the two facets of finding unknown objects in the object detection pipeline:

(1) Additional detection capabilities so that the unknown objects are not removed from the prediction head.

(2) Check for all of the predictions so that the model doesn't classify an OOD sample as ID sample.

For this task, we first visualize the object sample in latent representation hyper-space. We extract features at different layers from Faster-RCNN and visualize the data cluster using PCA and t-SNE. We also conduct a linear separability test as a qualitative measure of boundary separation between the data clusters, to decide which layer to use for feature extraction.

We fine-tune the faster-RCNN model from two dataset checkpoints: MS COCO and BigDetection. For the task of OOD detection, we propose a modified algorithm: using Mahalanobis distance as a measure for predicting confidence score for OOD classification. We also modify the algorithm from using RPN-based region proposals to using pre-trained models' predictions. Finally, we evaluate the models' predictions on sample set, before and after applying OOD detection algorithm, to evaluate its effectiveness for the task objectives we set above. Knowledge acquired over the course of this Master's program was successfully applied during this internship and proved instrumental in its success.

4.2. Future work

Here, we list some of the future research direction to pursue in OOD identification in object detection literature:

(1) Exploring STUD: Although the method explained in 1.4 is specifically proposed for object detection literature, with clever utilization of RPN proposals similar to our proposed method. it utilizes uniformly sampled video sequences, which are manually annotated. Hence, although this method works out-of-the-box for large-scale curated datasets, our experiments could not yield significant results on custom datasets. Due to limited resources, we stopped further exploration. It's worth noting that with recent development in the Segment Anything Model [**21**], a faster labelling process with little human intervention can utilized, which in turn can help with implementing this method.

(2) Exploring other object detection architectures: It's important to note that both our method and STUD are reliant on the two-stage detection nature of the algorithms. Although, the later improvement of our method requires predictions from two different models, making it compatible with other architectures as well. It would be an interesting problem to solve OOD detection for types of architectures (Single-shot detectors, anchor-free detectors, etc. refer section 1.3) in the domain, especially with the recent advent of Transformer-based models.

(3) Improving inference time and bench-marking: The proposed method requires one backward pass and multiple forward passes per proposed OOD bounding box, rendering it with a higher inference time per image. We believe this algorithm can set a baseline for further explorations in this domain. It is also worth noting that we need a better way to benchmark algorithms in this domain, compliant with the framework outlined in the OOD literature. Such a benchmark should also test for the two objectives we have outlined in this task. We believe that exploring a proper way to benchmark further development is crucial for this domain.

References

[1] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[2] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey, 2022.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[4] Likun Cai, Zhi Zhang, Yi Zhu, Li Zhang, Mu Li, and Xiangyang Xue. Bigdetection: A large-scale benchmark for improved object detector pre-training. *arXiv preprint arXiv:2203.13249*, 2022.

[5] Xuefeng Du, Xin Wang, Gabriel Gozum, and Yixuan Li. Unknown-aware object detection: Learning what you don't know from videos in the wild, 2022.

[6] Hiroto Honda. Digging into detectron 2- part 1, 2020.

[7] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, WENXUAN PENG, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. OpenOOD: Benchmarking generalized out-of-distribution detection. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[8] Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization, 2022.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[10] Ross Girshick. Fast r-cnn, 2015.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[12] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection, 2017.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[14] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2014.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[18] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection, 2019.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.