Features Synthesizing Network for Zero-Shot Object Detection

A Project Report

submitted by

ROHITHRAM R (EE16B031)

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

June 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Features Synthesizing Network for Zero-Shot Object Detection**, submitted by **Rohithram R**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Anurag Mittal
Research Guide
Professor
Dept. of Computer Science
IIT-Madras, 600 036

Prof. Kaushik Mitra
Department Co-Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

ACKNOWLEDGEMENTS

I wish to express my gratitude to my guide Prof. Anurag Mittal and my mentor Ashish Mishra (CS15D003), a PhD candidate from the CSE department, for their continued guidance and support throughout this project. I would also like to thank Prof. Kaushik Mitra for agreeing to be my department co-guide for this project.

I would also like to thank Pengkai Zhu, a Ph.D candidate in Electrical Engineering Department at Boston University, for clarifying few doubts regarding the implementation details in their paper, *Don't Even Look Once: Synthesizing Features for Zero-Shot Detection* [38]

Finally, I would like to thank Siddharth Nagar Nayak (EE16B073) for lending me GPU resources when needed for this project.

ABSTRACT

KEYWORDS: Zero-Shot Object Detection (ZSD); Zero-Shot Learning (ZSL); Generative Learning; Conditional Variational Auto-Encoder

The need for large scale object detection over thousands of object categories are gaining more importance each day, but collecting error-free or accurate ground truth annotations for that many classes is not scalable. And, the performance of trained object detection neural network is only limited to detection of seen categories (encountered while training) and consequently, the detection performance on unseen objects are generally degraded. The root of this problem can be traced back to the trained object detector's inability to assign higher confidence to the proposed bounding boxes of unseen object classes. Because, they are relegated as background by the model, in comparison to the bounding boxes of seen categories. Zero-Shot Object Detection (ZSD) addresses this problem by localizing and recognizing the unseen object categories during testing by often utilizing the external semantic information (word2vec, object attributes) of the seen and unseen classes.

With this in mind, in this project, we analyze and discuss some of the significant developments in ZSD over the recent years and their limitations. Then we analyze the results obtained from our implementation of the CVPR 2020 paper [38] from scratch, which addresses the problem discussed above by utilizing the generated visual features of unseen classes using a Conditional Variational Auto Encoder (CVAE). To validate the effectiveness of our implementation, we perform exhaustive ablative analysis.

Finally we conclude by posing some open questions to encourage further research, (i) applying ZSD to produce accurate annotations for large-scale datasets like Open Images Dataset [14]. (ii) incorporating background aware Latent Assignment Based (LAB) method [1] in our model for reducing false positive rates. (iii) extending our model from detection to Generalized Zero-Shot Recognition (GZSR) as an end-end learning task using Transformers [31, 2], which can make the model learn better discriminable visual features to achieve state of the art mAP.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ZSL	Zero-Shot Learning
ZSD	Zero-Shot Object Detection / Zero-Shot Detection
ZSR	Zero-Shot Recognition
GZSD	Generalized Zero-Shot Object Detection / Zero-Shot Detection
GZSR	Generalized Zero-Shot Object Recognition / Zero-Shot Recognition
DELO	Don't Even Look Once
YOLOv2	You Only Look Once v2
SSD	Single Shot Detector
RPN	Regional Proposal Network
GAN	General Adversarial Network
GTNet	Generative Transfer Network
VAE	Variational Auto-Encoder
CVAE	Conditional Variational Auto-Encoder
CNN	Convolutional Neural Network
FC	Fully Connected
TU	Test Unseen
TS	Test Seen
TM	Test Mix
AP	Average Precision
mAP	mean Average Precision
IOU	Intersection Over Union
tSNE	T-distributed Stochastic Neighbor Embedding
KLD	Kullback-Leibler Divergence
GPU	Graphics Processing Unit

CHAPTER 1

INTRODUCTION

Zero-Shot Learning (ZSL) method aims to solve a task without receiving any example of that task at training phase. The task of recognizing an object from a given image where there weren't any example images of that object during training phase can be considered as an example of Zero-Shot Learning task. To simply put, it allows us to recognize objects which the trained model have not seen before. First this methodology was applied to image classification problem to recognize unseen classes in test time which wasn't seen during training. This setting is popularly known as Zero-Shot Classification or Recognition (ZSR). Over the recent years, many developments has been done in this field using both traditional and generative methods [21, 32, 8]. This inspired the recent development of Zero-Shot Object Detection methods, which is just an extended setting of ZSL for object detection. We will see the motivation behind applying ZSL for object detection task in the next section.

1.1 Motivation

There has been remarkable developments in the world of object detection, where fully trained object detectors strive towards achieving higher average precision (AP), mean average precision rates (mAP) and higher detection speeds (frames per second) [27, 9, 26, 18, 28]. But it's performance is limited to only the annotated object categories available during training, and it's detection precision reduces significantly over unseen object categories owing to the lack of large-scale annotations, and low training samples per class. Datasets like PASCAL VOC [5] and MSCOCO [17] have only bounding box annotations for 20 and 80 classes respectively, this is much smaller in scale when compared to classification datasets like ImageNet [4] where ground truth class labels are available over thousands of classes, with a lot more samples per class.

And the task of annotating a visual scene is also very error-prone, where anything in the background can be assigned a label, hence even humans tend to miss out annotating few objects in the background. This makes it difficult for the models to learn as we are telling the model simultaneously that a certain object is both a positive (marked annotation correctly) and a negative (missed out annotation in the background of the same class) example. This can be observed even in standard datasets for object detection (PASCAL VOC, MSCOCO), where many of the background objects left unannotated. With these in mind, we need a method which can detect the similar or unseen object categories (we will address why 'similar' in later sections) by just utilizing the available training resources at hand.

Hence to overcome these challenges, the need of **Zero-Shot Object Detection (ZSD)** arises, which localizes and detects the unseen object categories when trained only on bounding box annotations of classes available during training time (seen classes). There has been development of several approaches over the recent years to solve the ZSD problem [36, 37, 38, 1]. Most of the approaches are based on the foundational concept which utilize the external class level semantic attributes or word embeddings of both seen and unseen classes to exploit the relationship between the visual and the semantic domain.

ZSD method reuses the existing object detectors of both the categories, single pass detectors such as YOLO [26], SSD [18] as well as two-stage detectors such as Faster-RCNN [28], F-RCNN [9], etc. A fully trained vanilla object detector is capable of producing bounding boxes for unseen categories, but it assigns very low confidence score or objectness score to it. This problem arises due to the confusing or relating unseen categories to the background in an image. Because of this, the model filters out or suppresses these bounding boxes which has low confidence scores as it only classifies top bounding box predictions into one of the seen classes. In zero-shot setting, there is no clear understanding of background class. In a standard supervised setting, the model learns to discriminate between background and the seen classes, whereas in ZSD setting, the background could also consist of unseen object categories or unannotated objects (missed annotations of seen objects). So the model learns to give low confidence scores to the unseen objects bounding box proposals when it tries to discriminate the background and the target objects (seen classes).

1.2 Overview

In this project, we mainly elaborate our understanding and discuss our results from our implementation of the algorithm presented in [38] from scratch. As discussed above, most of the approaches in ZSD take outputs of existing detectors given by Region proposal network (RPN) in Faster-RCNN [28], Darknet-19 in [27], and extract the predicted bounding boxes along with it's confidence scores as inputs into their system. This implies that their gains arise primarily from improved recognition (ZSR) rather than in placing bounding boxes of higher-confidence. In ZSD, the error associated with missed detection of unseen class has two components, one arises out of the low confidence object bounding proposal and the other component comes from the recognition (classification) of the given bounding box proposal into one of classes. As we mentioned above that the most of the recent developments focus on reducing the second error component. In contrast, [38] tries to improve localization performance by outputting high-confidence (objectness scores) bounding boxes for unseen objects to address the inherent precision-recall trade-offs that need rejection of background classes. Thus, it's main aim is to reduce the first error component. For this reason, following [38], in our implementation we mainly evaluate using class agnostic Average Precision (AP), which reflects overall detection performance on all the classes (discussed in section 3). It uses CVAE to generate visual features of unseen classes with the semantic vectors with an additional visual consistency loss (discussed in section 3). It then augments the training of the confidence predictor Conf(.), and uses sampling scheme (related to focal-loss [16]) to re-sample the proposed bounding boxes to prevent the background-foreground imbalance problem. We also vary the seen-unseen classes ratio to necessitate the improvement on object bounding boxes on unseen classes. We evaluate our implemented model in both the ZSD (only unseen classes in test set) and Generalized-ZSD (GZSD) setting where both seen/unseen objects exist in the test set, following [38].

Note, we train and evaluate our models only on PASCAL VOC dataset owing to the limited resources available during the course of the project. We denote our implemented model following [38], as **DELO** and the original model reported in [38] as **DELO[38]** for easier comparison of results. Our experiments following the algorithm in [38] evidently shows improvement in AP in the Test Mix (TM) configuration in the GZSD settings, i.e **36.6%** in vanilla YOLOv2 trained only on seen classes to **41.4%** in

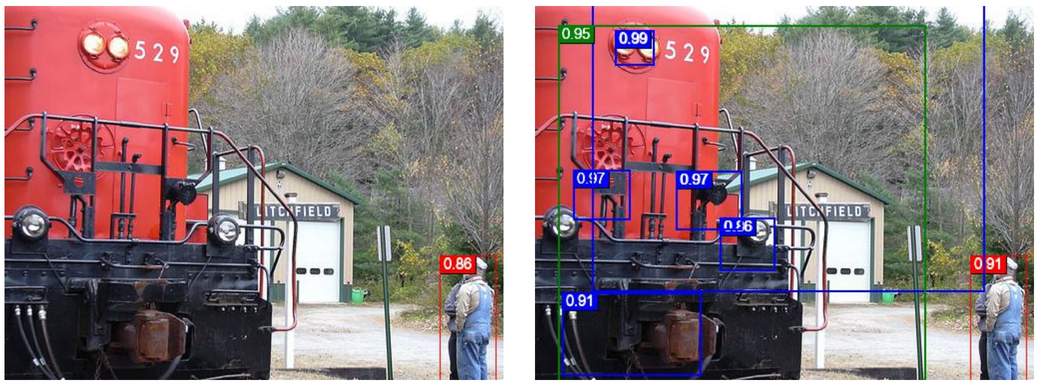

Figure 1.1: Example of detected unseen class in GZSD setting, after retraining confidence predictor with generated visual features of unseen classes. Our model DELO (Right) detects not only the unseen class train, but also the truly unknown or unannotated object classes like headlight, wheel, house/tree. Whereas YOLOv2 (left) only detects the seen class person. Refer to Table 4.1 for more details.

DELO at **0.6** confidence threshold on $10(\text{seen})/10(\text{unseen})$ classes split. Similar trends are also observed on $15/5$, $5/15$ data splits as well. We couldn't definitively compare our results with that of DELO[38], as [38] didn't mention the used confidence threshold to evaluate AP. Also since we train YOLOv2 following [27] from scratch, the backbone detector's performance (mAP/AP) plays a major role in how much %AP (absolute) is achieved in the end by DELO[38], as we build upon the pre-trained YOLOv2. But for the sake of completeness, we report our results along with results reported in [38]. We compare the relative increase in %AP from that of YOLOv2 to DELO to discuss the effectiveness of our model compared to results reported in [38]. Refer 4.1 for the complete evaluation results.

1.3 Contributions

To summarize, in this project

1) We analyze and discuss some of the significant developments in ZSD over the recent years, and it's limitations.

2) We analyze the results obtained from our implementation of the CVPR 2020 paper [38] from scratch. We then provide useful insights on the algorithm proposed by the paper, which addresses the problem discussed above by utilizing the generated visual features of unseen classes using a CVAE.

3) We then address why the performance gap between YOLOv2 and DELO isn't as

high as expected by analysing the DELO's detection of truly unknown or unannotated object classes which are counted as false positives.

4) We perform various experiments to validate the effectiveness of our implementation, such as experimenting on different sets of seen/unseen splits with varying number of similar or correlated unseen classes to seen classes, different choices of semantic embeddings, and different number of generated visual features of unseen classes.

5) Finally we conclude by posing some open questions to encourage further research, (i) applying ZSD to produce accurate annotations for large-scale datasets like Open Images Dataset (ii) incorporating background aware LAB method in DELO for reducing false positive rates. (iii) extending DELO from just detection to GZSR as an end-end learning task using Transformers, which can make the model learn better discriminable visual features to achieve state of the art mAP.

CHAPTER 2

LITERATURE REVIEW

Here, we briefly review the literature and discuss the significant contributions done by both the traditional approach [1] and generative approach to ZSD [36, 21], and finally we provide the necessary background for ZSD.

2.1 Related Work

2.1.1 Traditional approach to ZSD

Traditional approaches in Zero-Shot Learning (ZSL) has been extended to ZSR and ZSD setting in recent years. They rely on learning a common embedding space that aligns object features with class's semantic embeddings. Most of the traditional methods focus on mapping the visual features from predicted bounding boxes to the semantic embedding space [1, 23, 3, 29]. Specifically [1], maps or projects the visual features given by the Regional proposal networks like RPN in Faster R-CNN or Darknet-19 in YOLOv2, into semantic space by linear projection with a max-margin loss. Then a nearest neighbor search is done to predict the class of objects from unseen classes in test time [1]. Dealing with background is very important in ZSD as we discussed in chapter 1, that it creates confusion between the unseen classes and background classes. To address this problem, [1] proposed two background aware detectors, where they fix all of the unknown background classes into a single background category, second by doing iterative latent class (from a large vocabulary set) assignments to the background bounding boxes, in the aim of covering much wider visual concept. Since the number of seen classes is generally low in standard detection datasets like PASCAL VOC and MSCOCO, so projecting visual features to semantic domain during training become sparse in the semantic space, thus causing poor alignment between the visual and semantic modalities. To address this, [1] proposed a solution by densely sampling training classes using additional data.

2.1.2 Generative approach to ZSD

Mapping such high-dimensional visual feature to low-dimensional semantic space tends to cause the hubness problem due to the heterogeneity gap between these two spaces. Hence, the recent developments try to directly classify the objects using visual features itself, but utilising the semantic information of the seen classes to transfer the knowledge to unseen ones. This paves a way for generating visual features of unseen classes using GANs, CVAEs, and adversarial methods [36, 21, 38]

Specifically [36], proposed a GTNet, which consists of an Object Detection Module and a Knowledge Transfer Module, to tackle the ZSD problem. GTNet embeds a feature synthesizer in the Knowledge Transfer Module. The feature synthesizer learns to generate unseen class features, which will be further used to train an unseen category classifier for a pre-trained detector. It tries to address the very common problem of intra-class variance, and IOU variance existing in object detection. In order to synthesize features for each unseen class with both the intra-class variance and the IOU variance, [36] designs an IOU-Aware Generative Adversarial Network (IOUGAN) as the feature synthesizer which uses three GANs to generate class feature conditioned on class semantic embeddings, and a foreground feature, and finally a background feature conditioned on the corresponding class feature. Here each of the unit/GANs addresses intra-class variance, IOU variance, and background-awareness problems. Finally they add all three features and use it to classify only in the ZSD setting where they test images only with unseen object categories, which is much simpler compared to the more complex GZSD setting because, when we retrain the model on generated unseen visual features, it might develop a bias towards the unseen classes, and hence give better results in ZSD setting. But in GZSD setting, the test images contain both seen as well as unseen objects, thus model expected to perform poorly in this setting compared to other settings, as the model should simultaneously detect both the seen and unseen objects present. For this reason we have picked [38] to analyze and implement their proposed generative approach to ZSD.

2.1.3 Key Challenges

One of the major constraint in solving ZSD problem effectively is that the new unseen classes should be semantically related to the training classes. Which is a very important aspect in the model's capability of detecting unseen classes. Secondly, dealing with the background classes is pretty important in ZSD (to eliminate bounding box proposals which clearly do not contain any object of interest) because it affects the recall ability of the model to distinguish correctly between unseen classes and the background classes. Finally, since most of the standard detection datasets has only hundreds of classes, sparse sampling of classes during training leads to poorly learnt visual semantic representation, thus degrading the detection accuracy of unseen classes during test time and which led to the recent developments in generative ZSD.

2.2 Background

2.2.1 Zero-Shot Object Detection (ZSD)

The ZSR setting has been extended to the object detection problem, hence termed as ZSD. Over the past five years, a lot of research has been done on this topic, and the foundational concept is that they utilize the external class level semantic attributes or word embeddings of the classes to exploit the relationship between the visual and the semantic domain [30, 22, 11, 20, 19]. By mapping the visual features to semantic domain, it enables a way to recognize unseen classes. From naively doing a nearest neighbor search in the semantic domain consisting semantic attributes of both seen and unseen classes [1], to more complex methods using generative models such as conditional GANs and VAEs to synthesize features of unseen classes based on the corresponding class semantic attributes [21, 36, 38].

But unfortunately, there is no standard protocol, evaluation settings, datasets, or data splits (seen/unseen) have been followed by all the above methods [1, 23, 25, 15, 24, 37], and thus making it difficult for comparison of our model to existing methods. Specifically [1] uses recall@100 as the evaluation metric to report performance in the GZSD setting, owing to the very low mAP obtained. Recall@100 is a very loose metric to measure performance in GZSD setting because, in standard detection datasets like

PASCAL VOC, MSCOCO, there are only 5-10 foreground objects per image on average.So the unseen object bounding box proposals will easily come under the top 100 predictions resulting in unusually high gains. As we discussed in chapter 1 some ZSD methods [25, 13, 15] just focus on improving recognition of the extracted bounding box from the backbone detectors rather than improving the low confidence scores of the object bounding proposals of unseen classes by the pre-trained object detectors on seen classes. Few methods like [24] uses a transductive approach, where it exploits the unseen images during training, to evaluate the model in GZSD setting. Finally, datasets and the splits chosen by the existing methods are also not standard. For example, some uses ImageNet which has only one object per image, and some uses datasets which have clear to no background like F-MNIST [3]. Following these, result in unusually high gains as the dataset is simpler compared to complex datasets like PASCAL VOC, and MSCOCO which has multiple objects and a very complex background per image. Even some methods, use very high seen to unseen split [3, 24, 25] which again may result in higher gains, because the seen class features can easily be similar to or resemble unseen features due to the high split ratio (as the datasets generally consists of a cluster of classes belonging to a higher level category, for example, vehicle, indoor, etc.).

These various reasons made us pick [38] to analyze and implement and discuss our results. But, the software code for [38] is not publicly available, so we couldn't comprehensively validate our implemented model (DELO) to the original implementation (DELO[38]). Because, firstly there was no mention of the used confidence threshold to evaluate AP on different data splits, and secondly, since we train YOLOv2 from scratch, the backbone detector's performance (mAP/AP) plays a major role in how much %AP (absolute) is achieved in the end by DELO, as we build everything upon the pre-trained YOLOv2. But for the sake of completeness, we report our results along with results achieved by [38], and we compare the relative increase in %AP from that of YOLOv2 to DELO to discuss the effectiveness of our model compared to [38].

2.2.2 Object Detection

Object detection, is one of the important research topic in the world of computer vision. It's aim is to localize and recognize the objects of interest while suppressing the background objects. It generally comprises of two components, one which proposes object bounding boxes (regional proposals) and other is a classifier to classify each of those bounding boxes into an object category. Over the years, the main focus in this line of research has been towards achieving higher average precision, and higher detection speed (frames per second). Earlier approaches typically use the deep Convolutional Neural Network (CNN) [9, 28, 10, 34] for predicting bounding boxes per image, and a separate image classification CNN model to classify them. As the field started to develop significantly, the idea of doing detection in a single pass through deep CNN rather than two stage detectors was proposed [18, 26] this made end-to-end training feasible and have improved detection speed significantly compared to previous two-stage detectors. It also made real-time detection possible for the first time. In this project we use YOLOv2 as our backbone single-shot object detector following [38]. We chose YOLOv2 compared to other multi-stage detectors such as Faster-RCNN, R-FCNN, because they have much slower detection speed.

YOLOv2: Backbone Architecture

YOLOv2 is a single shot detection network which only uses CNNs from start to end to predict bounding boxes coordinates along with it's objectness score (confidence score), and softmax probabilities over target classes. It has more than 50 FPS detection speed, and is among the state of the art methods for object detection. It directly extracts visual features from image cells using Darknet-19 as the feature extractor, which is again a deep CNN which takes 416 x 416 image. It outputs a $F(I_m)$ feature map of size 13 x 13 x 1024 and uses a object predictor which contains 1 x 1 convolution layer to simultaneously produce a fixed set of bounding box proposals together with their associated confidence scores and softmax probabilities as mentioned above. YOLOv2 has five anchors to enable multiple object bounding box proposals with different aspect ratio per cell (prediction diversity) in the extracted feature map $F(I_m)$. So, the object predictor component has five bounding box predictors where each one of them outputs object locations (x, y), width and height of the bounding box of the assigned anchor, five confidence predictors, where each one is denoted as Conf(.) which gives objectness score $\hat{P}_{conf} \in [0, 1]$ associated with the predicted bounding box. It signifies if a bounding box is foreground (1) or background (0). Generally YOLOv2 has a classifier assigned to each anchor, but following [38] we train YOLOv2 only for the detection purpose and not for detection $+$ classification. Because training it for classification for just seen classes, will enable the model to learn to extract features from the image which are very discriminatory in nature, and close towards the seen classes. This may affect our model DELO's detection capability, and not likely to generalize well to unseen classes, but the effect could be minimal. Hence, the classification module (the classifiers) are detached from YOLOv2.

For GZSD these vanilla object detectors are ineffective in detecting unseen classes which it's not supervised during training, because it learns to suppress the image regions which are part of background. As we said earlier, in GZSD setting, the background consists of unseen objects as well. Thus making it difficult for the model to perform well in test images which contain both seen and unseen objects. Hence, to evaluate results of our implemented model, we will use YOLOv2 pre-trained only on images which has seen classes, as one of the baselines. We will validate our model by analyzing the relative increase in % AP achieved by YOLOv2 and by our DELO in the GZSD setting. This is the only comprehensive way of testing our effectiveness of our implementation following [38] for the reasons we mentioned in section 2.2.1.

Average Precision (AP) - Class Agnostic

We measure the overall detection performance, by computing AP over all classes in the dataset rather than a class specific AP as it's not possible to obtain class information, as we are just training it for detection and not classification as well. Therefore we do not compute mAP (mean average precision) or class specific AP to evaluate our models following [38]. We evaluate our model using PASCAL VOC 0.5 IOU - 11 point method to calculate AP. Where True Positive (TP) is defined as when a bounding box prediction has IOU > 0.5 with one of the ground truth objects. Where nGT is the no of ground truth objects, and Pred is the total no of predictions made by the model at a certain confidence threshold. Hence, we define precision (prec) and recall (rec) as follows:

$$\text{Prec} = \frac{\text{TP}}{\text{Pred}} \tag{2.1}$$

$$\text{Rec} = \frac{\text{TP}}{\text{nGT}} \tag{2.2}$$

AP is then computed as the mean precision at eleven uniformly spaced recalls $[0,\ 0.1,\ 0.2,\ ...]$

$$\text{AP} = \frac{1}{11} \sum_{r \in [0,0.1,0.2..]} \text{Prec}(\text{Rec} = r) \tag{2.3}$$

2.2.3 Word embeddings

Word embeddings are basically a form of distributed word representation that bridges the semantic relationship between words in a continuous vector form. Word embeddings are distributed representations of text in an n-dimensional space. In relation to the object detection, there have been developments in creating object level attributes present in an image, in order to improve the performance of the detection exploiting the semantic attributes along with the visual features. It enables us to describe unknown object categories (unseen classes), and report instance level atypical attributes of seen classes, and help the model to detect unseen object categories from pure textual description. Specifically [6] describes each object instance by 64 binary features/attributes. To illustrate, a sheep can either be represented via a word2vec which is learned by predicting co-occurring words, or the object attributes proposed in [6]. By following [6] a sheep is described as (has horn, has leg, has head, has wool) in a 64 dimension binary feature vector. We will analyse the effectiveness of following each of these methods in the ablative analysis section.

2.2.4 Conditional Variational Auto-Encoder (CVAE)

CVAE is a conditional generative model. It has encoder (E) and decoder (G) similar to VAE, which is based on Bayesian Inference. It's aim is to model the underlying probability distribution of data, enabling us to sample new data from that distribution. CVAE is different from VAE by having an additional input to condition upon to generate examples from specific class or anything which it is conditioned upon. In this project, following [38] we train CVAE to generate visual features of unseen classes, where we use semantic embeddings of each object category as conditional feature to condition upon. Thus, we could generate visual features of a specific unseen class with semantic embedding, which can't be done with the vanilla VAE. We will discuss CVAE in detail

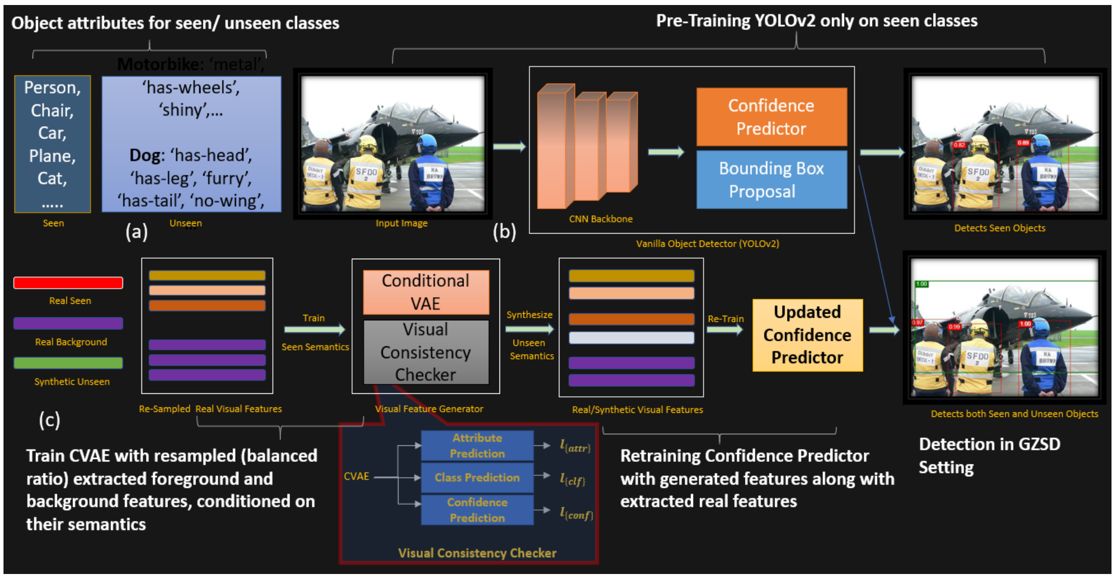
Object attributes for seen/ unseen classes
Pre-Training YOLOv2 only on seen classes
Person, Chair, Car, Plane, Cat, ...
Motorbike: 'metal', 'has-wheels', 'shiny',...
Dog: 'has-head', 'has-leg', 'furry', 'has-tail', 'no-wing',
Seen
(a)
Unseen
Input Image
(b)
CNN Backbone
Confidence Predictor
Bounding Box Proposal
Vanilla Object Detector (YOLOv2)
Detects Seen Objects
Real Seen
Real Background
Synthetic Unseen
Train
Seen Semantics
Conditional VAE
Visual Consistency Checker
Synthesize
Unseen Semantics
Re-Train
Updated Confidence Predictor
Detects both Seen and Unseen Objects
Re-Sampled Real Visual Features
Visual Feature Generator
Real/Synthetic Visual Features
(c)
Train CVAE with resampled (balanced ratio) extracted foreground and background features, conditioned on their semantics
CVAE
Attribute Prediction
Class Prediction
Confidence Prediction
$l_{\{attr\}}$
$l_{\{clf\}}$
$l_{\{conf\}}$
Visual Consistency Checker
Retraining Confidence Predictor with generated features along with extracted real features
Detection in GZSD Setting

Figure 2.1: This is the overview of the DELO network. (a) denotes the aPY object attributes proposed in [6], we use it as semantic embeddings for seen/unseen classes description following [38] (b) A vanilla detector trained only on seen objects which suppresses the confidence score of unseen objects confusing it with background, we use YOLOv2 as the object detector. It has 5 confidence predictors and bounding box proposal predictors assigned to each of it's five anchors. (c) We train the generator (CVAE in our case) with the resampled (balanced ratio) extracted foreground and background features and their semantics. Further, we generate unseen class visual features and retrain the confidence predictor module with all the seen/background/unseen features, and integrate it back to the object detector to detect unseen objects in GZSD setting. NOTE: Network template is inspired by [38]

in the next section.

CHAPTER 3

METHODOLOGY

As our goal is to implement the algorithm proposed in [38] effectively. In this section, we will elaborate the method to be followed to implement DELO with more nuanced and practical details which we thought were missed in [38]. And in addition to that, we also discuss few modifications we made in our implementation.

3.1 Problem Definition

We work on the standard object detection datasets, and we specifically cater or alter it to ZSD setting, where each of the image has multiple bounding box annotations with a class label for each of the objects present in it. So we define a training dataset with M images, with it's annotations and class labels as $D_{tr} = \{I_m, \{\text{Obj}_m^{(i)}\}_{i=1}^{N_m}\}_{m=1}^{M}$. Where $\{\text{Obj}_m^{(i)}\}_{i=1}^{N_m}$ is the collection of annotation and class label for an image I_m, and N_m is the number of objects present in the m^{th} image. Each object is associated with a bounding box annotation containing four values $(x,\ y,\ w,\ h)$ where $x,\ y$ is coordinates of the location and $w,\ h$ is the size of the bounding box, and a class label from $c \in C_{seen}$. Whereas the test images contains objects belonging to unseen classes i.e $c \in C_{unseen}$. We split our train and test partition such that $C_{seen} \cap C_{unseen} = \phi$. Our end goal is to detect all the foreground objects from both seen and unseen classes present in the image. We use semantic embedding S_c to train CVAE to generate the visual features of unseen classes. Thus we get semantic embeddings for all the classes including seen, unseen, and background $(c_{bg} = -1)$. We take $S_{-1} = 0$ for background classes.

3.2 Key Idea

As we discussed in section 1.2, our main objective is to reduce the error component by virtue of detection only, and not recognition. As we discussed in 1.1 that the vanilla detectors trained only on seen classes suppress the background objects, thus by unseen objects indirectly, which results in low confidence scores. This further lead to poor precision-recall rates for unseen objects in GZSD setting. So our main objective is to improve this bad precision-recall rates for unseen objects. And since anything in a visual scene can be labelled to some unknown object, the background bounding boxes are way larger in number compared to proposed foreground objects, hence reduces the precision of the detector. So, our second objective is to prevent this class imbalance problem by following re-sampling strategy proposed in [38] where we make the background/foreground samples ratio to 1. Lastly we conduct all our experiments and evaluation in GZSD setting which has both seen and unseen objects in the images for the reasons discussed in section $1, 2$.

To meet all these objectives, [38] proposed that re-training the confidence predictor with the generated visual features of unseen classes using CVAE along with the re-sampled seen and background features. Which would suppress the background objects and detect both the seen and unseen objects at a much higher confidence than before, as the re-trained model has learnt the visual features of seen, unseen and background separately, and thus distinguishes them easily.

As discussed above, our implementation following [38] has four stages, pre-training YOLOv2, re-sampling extracted seen and background features, visual feature generation, and confidence predictor retraining. Firstly, we train YOLOv2 just for detection after detaching the classifier module as we discussed in section $2.2.2$. (for training details refer section $4.1.4$

3.3 Foreground/Background Re-Sampling

After pre-training YOLOv2, we now extract visual features from the Darknet-19 (more details in section $4.1.4$). Our goal is to extract a collection of visual features representing the proposed foreground and background bounding boxes, and then re-sample it before re-training. Here we assume that cell feature of size 1024 in the $13 \times 13 \times 1024$ (hence, 169 cells per image) feature map $F(I_m)$, associated with the respective foreground or background bounding box is a reasonable visual representation of it as assumed in [38]. As we know that there are five confidence predictors associated with five anchor boxes per cell, and consequently the confidence predictors is associating to five bounding boxes proposed per cell. Those bounding boxes are probably at different locations, so their confidence predictions are different, although they are predicted on the same feature. Which means, the same feature, can be foreground to one anchor box and background for another, and potentially none (discard) having the same cell feature. Hence, we might encounter the same cell feature more than once in this extraction process, as the same cell feature could be viewed differently by each of 5 anchors. We represent our re-sampled dataset as D_{res}.

3.3.1 Foreground

For an image I_m, there are totally $169 \times 5 = 845$ bounding boxes in YOLOv2, and each cell has 5 bounding boxes associated with it, each one for the 5 anchors. We define a cell feature of an image I_m to be foreground w.r.t an anchor, if it's associated bounding box has maximum IOU > 0.5 with the ground truth objects for that image I_m and it's confidence score $\hat{P}_{conf} > 0.6$. We store $(f,\ a_{idx},\ \hat{P}_{conf},\ c)$ as a data point to our D_{res} dataset, where a_{idx} is the index of the anchor that bounding box with the maximum IOU is associated with and c is the class of the ground truth object which the bounding box has maximum IOU with. In our implementation, we store the anchor index explicitly as opposed to [38], for the reasons we will discuss in the section 3.5

3.3.2 Background

Similarly, a background feature to an anchor is defined when, it's associated bounding box has maximum IOU < 0.2 over the ground truth objects in the image I_m, and it's confidence score $\hat{P}_{conf} < 0.2$. We store $(f,\ a_{idx},\ \hat{P}_{conf},\ c_{bg})$ as a data point to our D_{res} dataset, where c_{bg} is the background class which is always set to -1. According to the re-sampling strategy , We only take k_m top smallest IOU background features, where K_m is the number of extracted foreground features.

3.4 Visual Feature Generation

After re-sampling of foreground and background features, we train the CVAE with the D_{res} dataset conditioned upon their class-specific embedding S_c, with the additional visual consistency loss functions proposed in [38] to provide more supervision and generate similar performing features as D_{res}. We add few extra details as compared to [38] which we think is not implicitly understood, to train the model easily. As we know that each feature is associated with an anchor index corresponding to it's bounding box, we train the CVAE separately on the feature groups. Each feature group is formed by combining all the extracted features which are associated to a particular anchor index, like this we do for all the anchor indices present in the D_{res} dataset. Hence, we train the CVAE each time separately on those feature groups corresponding to each of the anchor indexes present in the D_{res} dataset.

3.4.1 Conditional Variational Auto Encoder (CVAE)

As discussed in section 2.2.4, CVAE has an encoder E and a decoder G, and conditioned upon S_c. E with the parameter θ_E, takes a concatenated feature containing both the D_{res} feature f and the associated S_c, and outputs the distribution of the latent variable $z : P_E(z/f, S_c)$. The decoder G with the parameter θ_G generates the feature $\hat{f}$ given $(z,\ S_c)$. We find the optimal parameters $\theta_E,$ and θ_G simultaneously by optimizing CVAE loss which has two components, KL divergence between the encoder posterior and prior of latent variable z and reconstruction loss. l_{CVAE} is as given below:

$$l_{CVAE}(\theta_G, \theta_E) = \text{KL}(P_E(z/f, S_c)||p(z)) - \text{E}_{D_{res}}[\log P_G(f/z, S_c)] \tag{3.1}$$

where the KL divergence term's objective is to force the conditional posterior distribution approximates the true prior (unknown). To simplify the optimization, we use the re-parameterization trick followed in [12]

3.4.2 Visual Consistency Checker

This additional loss component has been proposed in [38], but doesn't have any theoretical validation. In our implementation we add this additional loss following [38], to make a fair comparison of our results with theirs. Main objective of this is to make the feature $\hat{f}$ be consistent with the extracted feature f in multiple aspects, such as $\hat{P}_{conf},$ class $c,$ and the associated S_c.

Confidence Consistency

So the objective here is to reconstruct features which outputs similar confidence score when given to pre-trained YOLOv2. So here we minimize the MSE Loss as given below:

$$l_{conf}(\theta_G) = \mathrm{E}_{D_{res}}|\hat{P}_{conf} - \mathrm{Conf}_{a_{idx}}(\hat{f})|^2 \tag{3.2}$$

where the $\mathrm{Conf}_{a_{idx}}$ is the confidence predictor associated with the corresponding anchor index of the feature f. We freeze the weights of the pre-trained confidence predictor while training the CVAE.

Classification Consistency

Similarly our objective is to make the reconstructed features more discriminatory and represents the same class c when given to a classifier Clf. Hence, we penalize the CVAE with l_{clf} is as follows:

$$l_{clf}(\theta_G) = \mathrm{E}_{D_{res}}[\mathrm{CE}(\mathrm{Clf}_{a_{idx}}(\hat{f}), c)] \tag{3.3}$$

where $c \in C_{seen} \cup \{-1\}$, and the $\mathrm{Clf}_{a_{idx}}$ is the classifier pre-trained only on the features corresponding to a_{idx}, and it's weights are frozen. It's trained with the weighted cross entropy loss to predict the class label associated with the feature f.

Attribute Consistency

The objective here is to reconstruct features which are coherent with semantic embeddings S_c of the feature f. So here we minimize the MSELoss as given below:

$$l_{attr}(\theta_G) = \mathrm{E}_{D_{res}}|S_c - \mathrm{Attr}_{a_{idx}}(\hat{f})|^2 \tag{3.4}$$

where the predictor $\mathrm{Attr}_{a_{idx}}$ is pre-trained on features corresponding to a_{idx} to predict the S_c associated with the feature f. We freeze the weights of it while training CVAE and use class weights to prevent the class imbalance problem.

Thus, we train the CVAE end-end by weighted addition of all these losses as follows:

$$\theta_G^*, \theta_E^* = \mathrm{argmin}_{\theta_G, \theta_E}\; l_{CVAE} + \lambda_{conf}.l_{conf} + \lambda_{clf}.l_{clf} + \lambda_{attr}.l_{attr} \tag{3.5}$$

where $\lambda_{[.]}$ are the weights assigned to each of the visual consistency losses. (refer training details in section $4.1.4$. Thus, after training, we generate D_{syn} dataset which has N_{seen} samples for each of the seen classes and N_{unseen} samples for each of the unseen classes. Thus a data point in D_{syn} looks like $(\hat{f}, 1, c)$ where $c \in C_{seen} \cup C_{unseen}$ and 1 is the confidence score assigned to the synthesised feature, as we consider the generated feature to be coming from a ground truth seen/unseen object.

3.5 Confidence Predictor Re-Training

Now, that we have D_{res} and D_{syn}, we re-train the confidence predictors. As we know that there are five confidence predictor associated with five anchor boxes, and consequently the confidence predictors is associated to five bounding boxes. Those bounding boxes are probably at different locations, so their confidence predictions are different, although they are predicted on the same feature. Which means, the same feature, can be foreground to one anchor box and background for another, and potentially none (discard). That is why we have to work on each anchor box separately.

We have trained all five confidence predictors in parallel, in contrast to [38], where they train each confidence predictor separately on their corresponding feature groups associated with that anchor. By training in parallel, we make the training more elegant and efficient. And since, theoretically there is nothing against following the parallel training method to the best of our knowledge. We use the anchor index of each feature to get the $\hat{P}_{conf}$ from the confidence predictor associated with that anchor to compute the MSE loss l as follows

$$l = \frac{1}{|D_{res}|} \sum_{f \in D_{res}} \sum_{a_{idx} \in f} |\text{Conf}_{a_{idx}}(f) - \hat{P}_{conf_{a_{idx}}}|^2$$

$$+ \quad \frac{1}{|D_{syn}|} \sum_{f \in D_{syn}} \sum_{a_{idx} \in f} |\text{Conf}_{a_{idx}}(\hat{f}) - 1|^2$$

where a_{idx} refers to the anchor index associated with the feature f, and $\hat{P}_{conf_{a_{idx}}}$, and $\text{Conf}_{a_{idx}}$ are the confidence score of the feature associated with that anchor, and confidence predictor associated with the anchor index respectively.

3.6 Implementation Details

We followed the same architecture mentioned in [38], for Clf, Attr, CVAE. The encoder E takes in a concatenated feature of dimension $1024 + 20$, where 1024 is the feature size and 20 is the dimension of the semantic embedding S_c used. The latent dimension is set to 50 and decoder G has input size of $50 + 20$ which is the sum of the dimensions of latent vector z and semantic embedding. And both E and G have hidden layer of size 128. Hence, they are basically a two fully connected layer network.

Clf(.) and Attr(.) are similarly chosen to be 2-FC networks with hidden dimension of size 256. We use $\lambda_{conf} = 1$, $\lambda_{clf} = 2$, $\lambda_{attr} = 2$, and generate $N_{seen} = 50$ and $N_{unseen} = 2000$ for every seen and unseen classes respectively. Refer section 4.1.4 for more training details.

CHAPTER 4

EVALUATION

4.1 Experimental Setup

We follow the evaluation protocol mentioned in [38], that to evaluate our implemented model DELO only in ZSD and GZSD settings. Where the test images contains only unseen objects in the former setting, and in the latter setting, test images has both seen and unseen objects, which is much more realistic and complex. We thus give less importance to pure unseen object detection results of [1, 25, 15] and also avoid the transductive setting followed in [24] as we discussed in the section 2.2.1.

4.1.1 Dataset

We only use PASCAL VOC dataset for the reasons discussed in section 2.2.1. It has 20 classes with 4 high level categories such as, person, vehicle, animal, and indoor, with indoor and vehicle combined having 13 classes, person and animal having 1 and 6 classes respectively. We also train and test our model DELO, on multiple seen-unseen split ratios, such as $15/5$, $10/10$, $5/15$ splits. In addition to what is done in [38], among each ratio, we have created different sets of unseen classes with varying no of similar classes to that of classes in the seen bucket. This is an important factor as we discussed in the background section 2.1.3, that unseen classes are assumed to be semantically or visually similar to the seen classes for ZSD to work. So we study the performance of our model DELO, on these different splits with varying low no of similar classes in the unseen bucket to a very high no of similar classes, and see the effect of high seen/unseen ratio to low seen/unseen ratio. We also observe that the person category is all alone, and don't have any similar classes, for this reason in most of our splits, we put person under seen split.

4.1.2 Test Data Configuration

Following [38], We test our model DELO on three test configurations, Test-Seen (TS), Test-Unseen (TU), and Test-Mix (TM). Test-Seen contains images from VOC2007 set which contains only seen objects. Test-Unseen contains images from train/val/test of VOC2007 and VOC2012 which has only unseen objects. Similar to Test-Unseen, Test-Mix contains images only which has both the seen and unseen objects. As discussed above, Test-Mix configuration is the GZSD setting, which is the most complex setting compared to the other two test configurations. We follow 0.5 IOU 11 point method to compute AP following [38] to evaluate our model DELO.

4.1.3 Semantic Information

Following [38], we use class-level semantic features as opposed to the object level attribute annotations from aPY [6] for the classes in PASCAL VOC. Because, firstly it's impossible to get object level attributes for unseen classes during test time to generate unseen visual features. Secondly by averaging them, we reduce the noise in the instance level attributes, thus resulting in better performance. Finally, the apY has given only object level attributes for a VOC 2008 dataset, and hence lot of objects in our train and test dataset don't have object level attributes. So, we take average of all the object level attributes for each of the 20 classes and we reduce the dimensionality from 64 (discussed in section $2.2.3$) to 20 using PCA to reduce the noise. We then normalize the semantic features between 0 and 1. We also use word2vec [20] as an alternative to study the importance of object attributes. Word2vec is of 300 dimensions, and is noisy. We compare the performance using each of these as the semantic embedding in our ablative analysis section.

4.1.4 Training Details

Pre-training YOLOv2

We have used PyTorch framework to code our implementation from scratch. We trained and tested our model DELO on a single NVIDIA GTX 1080Ti GPU on PASCAL VOC dataset. We followed [27] for training YOLOv2 just for detection on seen classes, where we trained it for 120 epochs for $15/5$ and $10/10$ splits, and 150 epochs for $5/15$ split, with a starting learning rate of 0.00025 and later adjust it in steps of $0.1,\ 10,\ 0.1$ after $10,\ 40,\ 80$ epochs respectively with SGD optimizer with 0.0005 decay and 0.9 momentum, and with a batch size of 32.

Extraction of feature-map

After the pre-training of YOLOv2, we extract the 13 x 13 x 1024 feature map $F(I_m)$ from the 29^{th} layer in YOLOv2. We take the output just after the linear activation before the batch-normalization and not after the leaky-relu activation, to avoid the manual scaling of the features when retraining the five confidence predictors. This enables us to directly feed the generated features from CVAE decoder directly, which uses linear activation as the output activation.

Training Attr, Clf and CVAE

For training Attr, and Clf, we scale the extracted features between 0 and 1, and train each of them for 100 epochs using ADAM optimizer with a learning rate of $1e-3$, and a learning rate decay of 0.5 every 15 epochs, with a batch size of 64.

For training CVAE, we train it for 100 epochs using ADAM optimizer with a learning rate of $1e-3$, learning rate decay of 0.75 every 15 epochs for $15/5$ and $10/10$ splits, and 200 epochs with a learning rate decay of 0.75 every 60 epochs for $5/15$ split, with a batch size of 256. We also used class-weights computed using $sklearn$ for all the loss functions used in training the above three models to prevent the class-imbalance problem caused by huge number of background samples compared to foreground samples per class.

Training CVAE can be tricky, we have used cyclic linear KL annealing rate for preventing KLD loss immediately going to zero, and make it converge to a non-zero number, so that CVAE doesn't generate random numbers losing all the latent information.

Method	Split	TU	TS	TM
YOLOv2[38]	5/15	36.6	85.6	30.0
DELO[38]		39.4 (2.8)	88.2 (2.6)	34.7 (4.7)
YOLOv2		25.5	81.2	17.6
DELO		**27.4** (1.9)	**82.5** (1.3)	**22.0** (4.4)
YOLOv2[38]	10/10	56.4	71.6	54.3
DELO[38]		61.3 (4.9)	73.5 (1.9)	59.6 (5.3)
YOLOv2		43.9	68	36.6
DELO		**47.3** (3.4)	**69.6** (1.6)	**41.4** (4.8)
YOLOv2[38]	15/5	55.3	75.3	53.6
DELO[38]		58.1 (2.8)	76.3 (1.0)	58.2 (4.6)
YOLOv2		45.6	70.4	44.3
DELO		**47.8** (2.2)	**71.1** (0.7)	**48.2** (3.9)

Table 4.1: Zero-shot detection evaluation results on the PASCAL VOC dataset for different seen/unseen splits. TU = Test-Unseen, TS = Test-Seen, TM = Test-Mix represents different data configurations. Overall average precision (AP) in $\%$ is reported. The highest AP (only among the APs of our implemented models i.e YOLOv2, DELO) for every setting is in bold. Note: YOLOv2[38], and DELO[38] are the results obtained in [38], and $(.)$ denotes the relative increase in AP from YOLOv2 to DELO

Retraining Confidence Predictor

As discussed in section 3.4, we train all five confidence predictors in parallel. We use weighted MSE Loss to prevent class imbalance problem caused by background features. We compute the sample weights by first computing class weights using $sklearn$, where we just take features from D_{res} to be an imaginary class 0 and D_{syn} to be an imaginary class 1, and then we assign these weights to all the features accordingly to get the sample weights.

We have used a batch-size of 256, and retrained it for 5 epochs with $2.5e-6$ learning rate.

4.2 Zero Shot Detection Evaluation

We tabulate the AP obtained by our implemented model DELO, and by it's corresponding pre-trained YOLOv2 on various seen/unseen splits in PASCAL VOC dataset in Table 4.1. Our main goal is to compare the relative increase of AP from that of YOLOv2

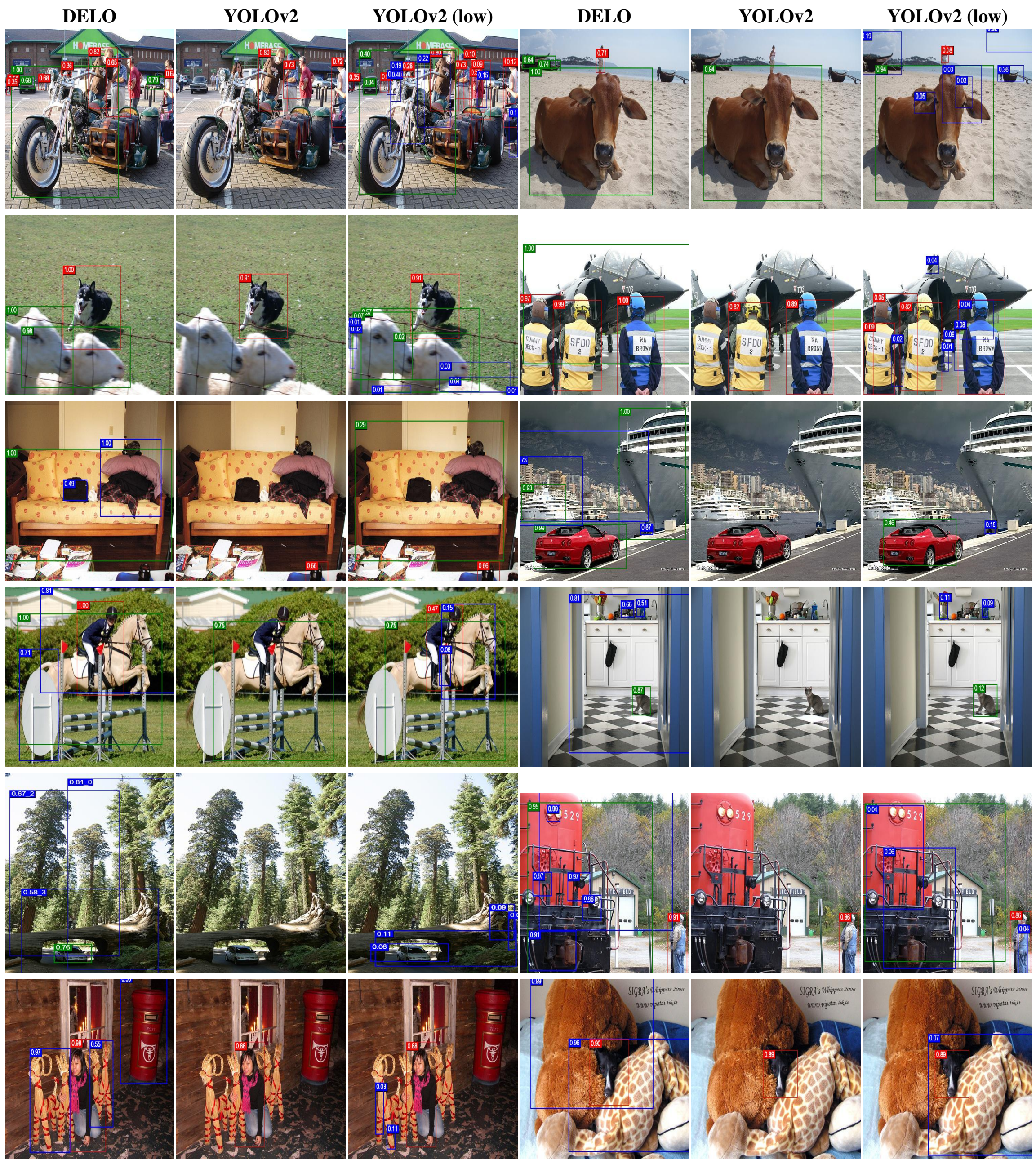

Figure 4.1: Visual examples of detections made by our DELO. Each triple shows: (from left to right) DELO detection results, pre-trained YOLOv2 detection results at the same confidence threshold as DELO, pre-trained YOLOv2 detection results at a much lower confidence threshold. The seen, unseen and unknown (error) classes are color-coded as red, green and blue. We observe that the implemented DELO detects unseen classes with a much higher confidence which are missed by YOLOv2 or in some cases constantly predicted with much lower confidence scores. And hence, pre-trained vanilla YOLOv2 suffers from significant detection errors when tried to detect unseen objects by lowering the confidence threshold. DELO also detects many truly unknown objects which are counted as false positives, refer section 4.2.1 for more details.

to DELO for each data split across Test configurations (TU, TS, TM). As we discussed in section 2.2.1, we can't fairly compare our performance with that reported in [38].

And we will give a detailed explanation as to why in the next section. But for the sake of completeness, we also report our results along with the results reported in [38], and are denoted as YOLOv2[38], and DELO[38].

4.2.1 Discussion on the outputs of DELO

Performance on Test Seen

From Table 4.1, we observe that both our pre-trained YOLOv2 and YOLOv2[38] perform very well on par to our DELO and DELO[38] in TS configuration compared to TU and TM. This is due to the fact that it's already been well optimized to detect only the seen objects in an image during the training, and as reported in [27] YOLOv2 is one of the state of the art methods in object detection which achieves 73.4% mAP on PASCAL VOC 2012 dataset. Hence, it achieves $81.2\%,\ 68\%,$ and 70.4% on $5/15,\ 10/10,\ 15/5$ splits respectively in the Test Seen (TS) configuration. We also observe that the performance by YOLOv2[38] is slightly better than our pre-trained YOLOv2 by $4.4\%,\ 3.6\%,$ and 4.9% on $5/15,\ 10/10,\ 15/5$ splits respectively in TS. This is because we have trained YOLOv2 from scratch, and due to different hyper parameter choices our performance is not as high as YOLOv2[38], this is one of the main reasons that make the comparison between our DELO and DELO[38] little unfair, as we aren't starting of with the similar performing baseline detector. We can also observe that the difference in AP achieved by our DELO and DELO[38], is significant in TU and TM configurations on all splits, but only slight difference in TS configuration. That's because, detecting in only seen objects in test seen images are relatively easier compared to other settings, as the model is already pre-trained on the seen-partition of the dataset which has only seen objects. Consequently, YOLOv2 is a strong baseline to compare against particularly in TS category, and also in cases where seen/unseen ratio is increasing, most of it's performance in TM can be coming from the seen classes detection. Because in cases like $15/5$ split, it's more likely that unseen visual features resemble samples from seen classes, and consequently don't require confidence predictor retraining for better detections. Therefore, in these Test Mix configurations, YOLOv2 is expected to perform better for this reason. To sum it up, pre-trained YOLOv2 object detectors that are optimized only over seen objects during training and not unseen objects are still capable of localizing unseen objects but with a much lower confidence score (objectness score) compared to our DELO, this can be observed in the 3^{rd} and 6^{th} column in the Figure 4.1

Effectiveness of retraining with generated features

We observe that both DELO, and DELO[38] consistently performs better than the pre-trained YOLOv2, and YOLOv2[38] in all the data splits, and test configurations. This validates the effectiveness of the retraining the confidence predictor with the generated features using CVAE. Thus, DELO leverages visual features from unseen, and a balanced seen and background bounding boxes after re-sampling to prevent class imbalance problem, it learns to better discriminate the seen/unseen from that of background features. Hence, we observe an average increase in AP of (2.5%, 1.2%, and a significant 4.37%) by our DELO, compared to (3.5%, 1.83%, 4.87%) by DELO[38] across all the splits in TU, TS, and TM configurations respectively. This trend followed in the relative increase in AP in TU, TS, and TM configuration is expected because, we typically expect a lower AP for Test-Mix compared to other two configurations. And we intuitively expect to observe highest AP in Test-Seen as the model DELO is pre-trained to detect only seen objects as we discussed in the above section, and the second highest in Test-Unseen, as the model may develop a bias towards the unseen classes after retraining the confidence predictors, and thus may perform a bit poorly on Test-Mix where it has to detect both seen and unseen classes, hence resulting in lowest AP in the Test-Mix. Thus, we observed the similar trend being followed in the average relative increase in AP achieved both by DELO and DELO[38] across all the splits in the TU, TS, and TM configurations respectively.

For the reasons we mentioned in the above section, we might not be able to get the absolute AP achieved by DELO[38] as we observe in Table 4.1, owing to the fact the we started off with a not so similar performing baseline detector. But, we observe that the average relative increase in AP from YOLOv2 to DELO mentioned above is of the similar range as the average relative increase achieved by DELO[38]. Hence, our implementation is effective.

Effect of the False Positives on the Performance Gap between DELO and YOLOv2

We observe that the average performance gain from YOLOv2 to DELO in TU and TM configurations is only (2.5%, 4.37%) and is not much as we expected it to be after observing the accurate detections of unseen classes made by DELO which were missed by YOLOv2 in Figure 4.1. This is not surprising because, DELO also detects truly unknown or unannotated objects along with artificially separated unseen classes, w.r.t PASCAL VOC. This can be noticed in the Figure 4.1, were DELO detects truly unknown objects such as trees, toy-horses, teddy bears, headlights, utensils, buildings, pillows, and bags along with the unseen classes such as motorbike, car, cow, boat, sheep, aeroplane, sofa, cat, and train. We observe that these detected unknown objects have similar attributes/traits to the seen/unseen classes. For example, (tree, potted plant), (teddy bear/pillow, sofa), (toy-horse, horse), (utensil, bottle), (headlight, motorbike/car) are some of the similar pairs we noticed in the Figure 4.1. We believe that re-training with generated visual features of unseen/seen classes using their semantic attributes, confuses the model into detecting unknown objects with similar attributes. Which also shows that DELO isn't extracting much information from the background objects to precisely distinguish unseen from other unknown classes. (we have discussed this issue in future works section 4.4.2)

Unfortunately, these cases cannot be quantitatively measured. Because of the lack of ground truth annotations for many unknown objects in PASCAL VOC. Therefore, the detections made by DELO are counted as false positives, hence lowering the overall precision. Thus, DELO's performance gain can be expected to be larger than YOLOv2 in a much large-scale dataset such as Open Images Dataset [14]

NOTE: There were no false positives reported in the detections made by DELO[38] in [38], which is very surprising, and we think that they have filtered out the false positives.

Robustness of DELO to different seen/unseen splits

We observe that performance by both DELO and DELO[38] varies significantly for different splits of TM configuration since the dataset is of a smaller scale (only 20 classes overall and objects per class is limited). And the number of unseen classes is

Method	ZSD	GZSD
YOLOv2	4.3	9.4
DELO	**5.1**	**12.2**

Table 4.2: ZSD and GZSD performance evaluated with mAP on the PASCAL VOC dataset $10/10$ split to compare the performance of our implemented models, YOLOv2 and DELO.

also changing. AP achieved in $5/15$ split is the least observed because, the model has very few seen classes and data samples to learn from to generalize better to generate features for 15 unseen classes. So the lower absolute performance can be due to this reason. But we observe that the DELO's performance is still superior compared to the pre-trained YOLOv2.

Generalized ZSR (GZSR)

As we discussed in section 1.2, our main goal is to improve error component arising by virtue of detection and not the recognition following [38]. We have focused mainly on detection aspect of GZSD. But for the completeness, we have trained a 2 fully-connected (FC) neural network as a classifier to classify the predicted bounding boxes (features associated with the bounding boxes) into one of the 20 classes in PASCAL VOC. According to the literature, GZSR is a very hard problem in its nature for PASCAL VOC dataset with poor accuracy [33]. And many of the recent developments in GZSR report recall@100 as the evaluation metric owing to the purportedly low mAP [1], which gives unusual high gains as we discussed in section 2.2.1. Hence, for this reason we chose to report only mAP. We can observe in Table 4.2 that, the mAP improves from YOLOv2 to DELO in both ZSD and GZSD settings. Here we don't have any reference model to compare against as [38] reported mAP only on MSCOCO dataset and not on PASCAL VOC. This low mAP can be attributed to the fact that we didn't fine-tune the classifier or choose relevant number of features to train it on. Also generally YOLOv2 updates the entire network while training to for both detection and classification task to achieve a good mAP i.e it follows an end-end learning method. But in our case, the feature extractor is fixed which has been optimized to produce features to just perform detection, thus the features learnt are not that discriminatory as in the case of a typical detection + classification YOLOv2 model. But the effect of this may be

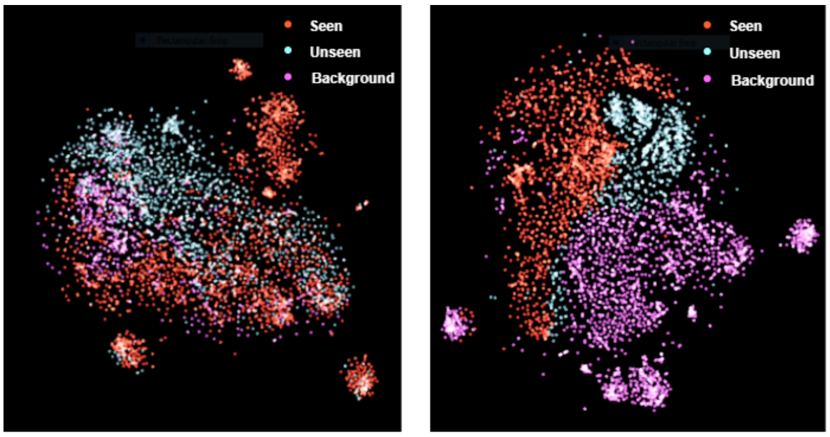

Figure 4.2: tSNE visualization comparison between extracted visual features from pre-trained YOLOv2 (left) with that from DELO (right) on $10/10$ split Test-Mix dataset. We observe that the foreground (seen and unseen) and background are more separable after retraining the confidence predictor with the generated features of unseen classes (DELO)

minimal. And also most of our bounding boxes for both seen and unseen classes are indeed correct with a very high confidence score compared to YOLOv2 as we can observe from the Figure 4.1. Hence, in future, we intend to combine our DELO detector with a good ZSR model or use an end-end GZSR framework using Transformers[31] (discussed in section $4.4.3$ as part of the future work) than a naive 2 FC classifier to achieve reasonable mAP.

4.3 Ablative Analysis

4.3.1 Effect of Retraining with Generated Visual features

To further evaluate the effect of the confidence predictor retraining with the generated visual features of unseen classes along with the re-sampled seen, and background class features. And its impact on the model's capability of accurately assigning a visual features as seen/unseen/background class for improved detection. We have used the tSNE dimensionality reduction technique to visualize the manifold of the extracted visual features from pre-trained YOLOv2 and DELO on the $10/10$ split's Test Mix dataset. We observe in Figure 4.2 that the seen, unseen classes are much more separable from background class when using DELO compared to YOLOv2, where background and unseen classes are mixed which affects the model recall and precision rate. Thus, our imple-

Split	AP(%)
15/15-1	47.8
15/15-2	45.6
10/10-1	47.3
10/10-2	44.1
10/10-3	33.7
5/15-1	27.4
5/15-2	23.5

Table 4.3: Seen/Unseen correlation comparison on the PASCAL VOC dataset. Where the number of correlated seen and unseen classes in the set, decreases with increase in the suffix attached to all the split names $(15/5,\ 10/10,\ 5/15)$. We observe that AP on Test Unseen decreases with the decrease in the number of similar or correlated seen and unseen classes in each of the splits.

mented model DELO, is effective and separates the unseen classes from the background better than the YOLOv2, hence, leading to the observed relative increase in AP from YOLOv2 to DELO. This ablation study hasn't been done in [38], hence not comparable with our results.

4.3.2 Effect of Seen/Unseen Classes Correlations

As we discussed in section $2.1.3$, one of the key challenges for effectively generating visual features given it's semantic embedding for unseen classes are constrained by unseen classes correlation with that of seen classes. It is expected for a model to not perform well when all the chosen unseen classes are not similar to any one of the seen classes. Which makes it difficult for the CVAE to generalize the learnt representation/distribution using seen classes, to sample visual features of unseen classes. So, with the goal to study the effect of the correlation between the seen and unseen classes semantically, on the detection performance of DELO on the unseen classes. We have trained and evaluated our DELO on different sets of unseen classes consisting of decreasing number of similar unseen classes to that of the seen classes present in that set. We have done this for all the three splits and have tabulated the AP on Test Unseen configuration in the Table 4.3. We have constructed 2 different sets for $15/5$ split namely $15/5-1,\ 15/5-2$ with the number of similar unseen classes decreasing with the increasing suffix attached to the split name. Similarly we have constructed three sets for $10/10$, and two for $5/15$. These sets follow the basic principle that, number of visually

Prototypes	AP (%) - Test Unseen (TU)
Object Attributes (20 dim)	**47.3**
Object Attributes (64 dim)	44.8
Word2Vec (20 dim)	45.2
Word2Vec (300 dim)	41.9

Table 4.4: Semantic prototype comparison on PASCAL VOC $10/10$ split data. Highest AP is in **bold**

or semantically similar classes appear in both the seen and unseen category separately, for a set to have higher correlation between seen and unseen classes. Similarly to have a lower correlation, the similar classes are assigned only to either seen or unseen category. To illustrate this with an example, in $10/10 - 1$ split, which has the highest correlated classes in seen and unseen, the classes present in vehicle category of PASCAL VOC i.e motorbike, car, bus, train are separated and equally put in seen and classes category. So DELO generalizes better on $10/10 - 1$ split, and achieves 47.3% AP, while in $10/10 - 3$ split it achieves only 33.7% AP, where all of vehicle category classes are put in seen category, hence having the least correlated seen and unseen classes.

4.3.3 Effect of various Semantic Prototypes and it's Dimensionality

To study the effect of the choice of the semantic embeddings used to generate the visual features, along with the effect of the role played by dimensionality of the embeddings. We have trained and evaluated our DELO on both the aPY object attributes [6] with 64 dimensions, and after it has been reduced from 64 to 20 as we discussed in section $2.2.3$. Similarly with the word2vec which is of 300 dimensions, proposed in [20] and on the reduced word2vec which is of 20 dimension following the same method discussed in $2.2.1$. We observe in Table 4.4 that, DELO trained with object attributes with 20 dimensions perform better than all the other prototypes with a 47.3% AP. We also observe that the dimensionality reduction in both the object attributes and word2vec is effective and improves the AP by 2.5%, and 3.3% respectively. We also note that word2vec with 300 dimensions has the least AP. Because, it is highly noisy and has too much information unnecessary to the visual features domain, which makes the CVAE poorly converge and not generalize better.

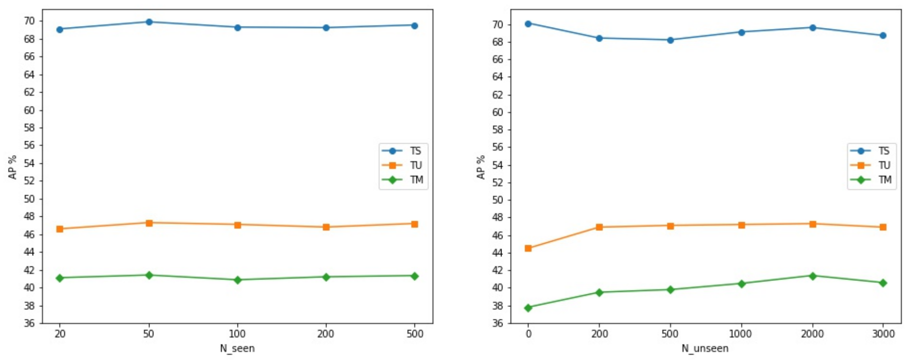

Figure 4.3: Performance when different N_{seen} (left) and N_{unseen} (right) has been used on $10/10$ split of PASCAL VOC. TU = Test-Unseen, TS = Test-Seen, TM = Test-Mix. We observe that the optimal values are $N_{seen} = 50$ & $N_{unseen} = 2000$

4.3.4 Effect of number of Generated Samples

To understand how many generated samples of unseen and seen classes are required to effectively retrain the confidence predictor. We have retrained and evaluated our DELO with different samples of size N_{seen} and N_{unseen}. We study the effect of N_{seen} by changing it from $[20,\ 50,\ 100,\ 300,\ 500]$ with N_{unseen} fixed at 2000. Similarly we change N_{unseen} from $[0,\ 200,\ 500,\ 1000,\ 2000,\ 3000]$ with N_{seen} fixed at 50. The corresponding AP obtained in TU, TS, and TM configurations are plotted in the Figure 4.3. We observe that, there is a significant drop in AP in TU and TM configuration, at $N_{unseen} = 0$, which is expected as the model is not learning anything about unseen classes, and now it's just performing similar to pre-trained YOLOv2. We also observe that the effect of N_{seen} on the performance is very minimal as it closely resembles D_{res} distribution. But a slight increase in AP can be observed in TS at $N_{seen} = 50$, which might be due to a number of seen samples the model requires to stabilize or to preserve it's learned weights for detecting seen classes, as we are simultaneously retraining it with unseen visual features. Finally, we see that AP in TU, and TM increase till $N_{unseen} = 2000$ after which it saturates. **Note**: We see that the most of the gain or increase in AP in TU and TM is observed just after retraining it with $N_{unseen} = 200$, thus very small number of generated unseen visual features are enough to improve the model's detection performance significantly. This is due to the ability of pre-trained YOLOv2 to already localize the unseen objects with just training on the seen partition. Hence, by retraining it with a small enough unseen visual features is enough to make it detect the unseen objects at a much more higher confidence score.

4.4 Possible Future Work

4.4.1 Annotation of large-scale datasets

As we discussed in section 4.2.1, that the DELO also detects truly unknown or unannotated objects along with artificially separated unseen classes, w.r.t PASCAL VOC. Utilizing this behaviour, we might be able to use DELO for predicting/generating missed annotations in the large scale dataset with noisy attributes like the Open Images Dataset [14]. Which has a lot of objects missed in the ground truth annotations, as the labelling is done by DNN + human, unlike trademark object detection datasets which have human level annotations like PASCAL VOC or MSCOCO. We leave the detailed analysis for future work.

4.4.2 Background Aware Latent Assignment Based ZSD for DELO

To extract more information from background region and cover wider visual concepts, than squishing all background classes to 0 in the embedding space as done in [38]. We can follow the Latent Assignment Based (LAB) method proposed in [1] to spread the background bounding boxes in embedding space, and then train the CVAE, iteratively, and finally generate the unseen visual features. We believe that, this will give lesser room for the model to confuse between an actual unseen class and an unknown class having similar attributes (which is now learnt to be a background object by the model after iterative training following LAB method). Hence, it reduces the overall false positive rate, and enhances the Average Precision (AP).

4.4.3 Transformer based End-End Zero-Shot Object Recognition

As we observed in section 4.2.1 that, the DELO + basic classifier performs very poorly in recognizing the seen/unseen classes in the GZSD setting. Hence, we believe that extending DELO from just training for detection to GZSR as an end-end learning framework, can make the model learn better discriminable visual features to achieve state of the art mAP.

Recently, Transformer based End-End Object Detection has been proposed in [2], where it removes the need of Non-Maximum Suppression and Anchors which are used in traditional object detectors. Transformers are inherently an encoder-decoder framework which was introduced in [31]. There also have been usage of Transformers for Zero-Shot Learning in recent years. Specifically [39], use it for multi-attention based localization of objects. It mainly specializes in multi-modal learning [35, 7], and we know that ZSD is also a multi-modal learning problem between visual and semantic domain. Hence with all these recent developments in mind, we believe that Transformers can be used to solve the Generalized Zero-Shot Object Recognition (GZSR) problem in a more elegant and in an end-end learning manner. However, we leave the detailed analysis for future work.

CHAPTER 5

CONCLUSION

Vanilla object detectors trained only on seen classes are capable of localizing unseen objects, but with a very low confidence score. To overcome this, and improve the localization and detection performance on novel unseen object categories in GZSD settings, we have followed the algorithm proposed in [38], and implemented it. We validated our implementation by comparing the relative increase or gain in AP from YOLOv2 to DELO, with that of the gain reported in [38]. We have also reasoned out why the performance gap between YOLOv2 and DELO isn't as high as expected, by analysing the DELO's detection of truly unknown or unannotated object classes which are counted as false positives. We have also conducted many experiments with different seen/unseen splits, different choices of semantic embeddings, and with different number of generated visual features. Our experiments, showed that the seen/unseen classes correlation is a very important factor in determining the performance gain from YOLOv2 to DELO in TU and TM configurations. Which is also a very major challenge going forward in the field of ZSD to localize and recognize a truly unknown object which are not related to the classes seen during training. Our experiments also showed that very small number of unseen generated samples are enough to re-train the baseline detector to get a significant performance gain. There are several areas for improvements such as incorporating background aware LAB method in DELO for reducing false positive rates, and extending DELO from just training for detection to GZSR as an end-end learning task using Transformers, which can make the model learn better discriminable visual features to achieve state of the art mAP. We leave the detailed analysis for future work.

REFERENCES

[1] **Bansal, A.**, **K. Sikka**, **G. Sharma**, **R. Chellappa**, and **A. Divakaran**, Zero-shot object detection. *In Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[2] **Carion, N.**, **F. Massa**, **G. Synnaeve**, **N. Usunier**, **A. Kirillov**, and **S. Zagoruyko** (2020). End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*.

[3] **Demirel, B.**, **R. G. Cinbis**, and **N. Ikizler-Cinbis** (2018). Zero-shot object detection by hybrid region embedding. *arXiv preprint arXiv:1805.06157*.

[4] **Deng, J.**, **W. Dong**, **R. Socher**, **L.-J. Li**, **K. Li**, and **L. Fei-Fei**, ImageNet: A Large-Scale Hierarchical Image Database. *In CVPR09*. 2009.

[5] **Everingham, M.**, **S. M. A. Eslami**, **L. Van Gool**, **C. K. I. Williams**, **J. Winn**, and **A. Zisserman** (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, **111**(1), 98–136.

[6] **Farhadi, A.**, **I. Endres**, **D. Hoiem**, and **D. Forsyth**, Describing objects by their attributes. *In 2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.

[7] **Gao, P.**, **H. You**, **Z. Zhang**, **X. Wang**, and **H. Li**, Multi-modality latent interaction network for visual question answering. *In Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[8] **Gao, R.**, **X. Hou**, **J. Qin**, **L. Liu**, **F. Zhu**, and **Z. Zhang**, A joint generative model for zero-shot learning. *In Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[9] **Girshick, R.**, Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision*. 2015.

[10] **Girshick, R.**, **J. Donahue**, **T. Darrell**, and **J. Malik** (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, **38**(1), 142–158.

[11] **Joulin, A.**, **E. Grave**, **P. Bojanowski**, and **T. Mikolov** (2016). Bag of tricks for efficient text classification. arxiv preprint arxiv: 160701759.

[12] **Kingma, D. P.** and **M. Welling** (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[13] **Kumar Verma, V.**, **G. Arora**, **A. Mishra**, and **P. Rai**, Generalized zero-shot learning via synthesized examples. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[14] **Kuznetsova, A.**, **H. Rom**, **N. Alldrin**, **J. Uijlings**, **I. Krasin**, **J. Pont-Tuset**, **S. Kamali**, **S. Popov**, **M. Malloci**, **A. Kolesnikov**, **T. Duerig**, and **V. Ferrari** (2020). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*.

[15] **Li, Z.**, **L. Yao**, **X. Zhang**, **X. Wang**, **S. Kanhere**, and **H. Zhang**, Zero-shot object detection with textual descriptions. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33. 2019.

[16] **Lin, T.-Y.**, **P. Goyal**, **R. Girshick**, **K. He**, and **P. Dollár**, Focal loss for dense object detection. *In Proceedings of the IEEE international conference on computer vision*. 2017.

[17] **Lin, T.-Y.**, **M. Maire**, **S. Belongie**, **J. Hays**, **P. Perona**, **D. Ramanan**, **P. Dollár**, and **C. L. Zitnick**, Microsoft coco: Common objects in context. *In European conference on computer vision*. Springer, 2014.

[18] **Liu, W.**, **D. Anguelov**, **D. Erhan**, **C. Szegedy**, **S. Reed**, **C.-Y. Fu**, and **A. C. Berg**, Ssd: Single shot multibox detector. *In European conference on computer vision*. Springer, 2016.

[19] **Mikolov, T.**, **K. Chen**, **G. Corrado**, and **J. Dean** (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[20] **Mikolov, T.**, **I. Sutskever**, **K. Chen**, **G. S. Corrado**, and **J. Dean**, Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*. 2013.

[21] **Mishra, A.**, **S. Krishna Reddy**, **A. Mittal**, and **H. A. Murthy**, A generative model for zero shot learning using conditional variational autoencoders. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018.

[22] **Pennington, J.**, **R. Socher**, and **C. D. Manning**, Glove: Global vectors for word representation. *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

[23] **Rahman, S.**, **S. Khan**, and **N. Barnes** (2018). Polarity loss for zero-shot object detection. *arXiv preprint arXiv:1811.08982*.

[24] **Rahman, S.**, **S. Khan**, and **N. Barnes**, Transductive learning for zero-shot object detection. *In Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[25] **Rahman, S.**, **S. Khan**, and **F. Porikli**, Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts. *In Asian Conference on Computer Vision*. Springer, 2018.

[26] **Redmon, J.**, **S. Divvala**, **R. Girshick**, and **A. Farhadi**, You only look once: Unified, real-time object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[27] **Redmon, J.** and **A. Farhadi**, Yolo9000: better, faster, stronger. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[28] **Ren, S.**, **K. He**, **R. Girshick**, and **J. Sun**, Faster r-cnn: Towards real-time object detection with region proposal networks. *In Advances in neural information processing systems*. 2015.

[29] **Shigeto, Y.**, **I. Suzuki**, **K. Hara**, **M. Shimbo**, and **Y. Matsumoto**, Ridge regression, hubness, and zero-shot learning. *In Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015.

[30] **Szegedy, C.**, **S. Ioffe**, **V. Vanhoucke**, and **A. A. Alemi**, Inception-v4, inception-resnet and the impact of residual connections on learning. *In Thirty-first AAAI conference on artificial intelligence*. 2017.

[31] **Vaswani, A.**, **N. Shazeer**, **N. Parmar**, **J. Uszkoreit**, **L. Jones**, **A. N. Gomez**, **Ł. Kaiser**, and **I. Polosukhin**, Attention is all you need. *In Advances in neural information processing systems*. 2017.

[32] **Xian, Y.**, **T. Lorenz**, **B. Schiele**, and **Z. Akata**, Feature generating networks for zero-shot learning. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[33] **Xian, Y.**, **B. Schiele**, and **Z. Akata**, Zero-shot learning-the good, the bad and the ugly. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[34] **Xu, H.**, **X. Lv**, **X. Wang**, **Z. Ren**, **N. Bodla**, and **R. Chellappa**, Deep regionlets for object detection. *In Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[35] **Yu, J.**, **J. Li**, **Z. Yu**, and **Q. Huang** (2019). Multimodal transformer with multi-view visual representation for image captioning. *IEEE Transactions on Circuits and Systems for Video Technology*.

[36] **Zhao, S.**, **C. Gao**, **Y. Shao**, **L. Li**, **C. Yu**, **Z. Ji**, and **N. Sang** (2020). Gt-net: Generative transfer network for zero-shot object detection. *arXiv preprint arXiv:2001.06812*.

[37] **Zhu, P.**, **H. Wang**, and **V. Saligrama** (2019). Zero shot detection. *IEEE Transactions on Circuits and Systems for Video Technology*, **30**(4), 998–1010.

[38] **Zhu, P.**, **H. Wang**, and **V. Saligrama**, Don't even look once: Synthesizing features for zero-shot detection. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[39] **Zhu, Y.**, **J. Xie**, **Z. Tang**, **X. Peng**, and **A. Elgammal**, Semantic-guided multi-attention localization for zero-shot learning. *In Advances in Neural Information Processing Systems*. 2019.