Don’t Even Look Once: Synthesizing Features for Zero-Shot Detection

Pengkai Zhu† Hanxiao Wang‡,† Venkatesh Saligrama†
† Electrical and Computer Engineering Department, Boston University
‡ Onfido ltd, London, UK
† {zpk, hxw, srv}@bu.edu, ‡ hanxiao.wang@onfido.com

Abstract

Zero-shot detection, namely, localizing both seen and unseen objects, increasingly gains importance for large-scale applications, with large number of object classes, since, collecting sufficient annotated data with ground truth bounding boxes is simply not scalable. While vanilla deep neural networks deliver high performance for objects available during training, unseen object detection degrades significantly. At a fundamental level, while vanilla detectors are capable of proposing bounding boxes, which include unseen objects, they are often incapable of assigning high-confidence to unseen objects, due to the inherent precision/recall tradeoffs that requires rejecting background objects. We propose a novel detection algorithm “Don’t Even Look Once (DELO),” that synthesizes visual features for unseen objects and augments existing training algorithms to incorporate unseen object detection. Our proposed scheme is evaluated on PascalVOC and MSCOCO, and we demonstrate significant improvements in test accuracy over vanilla and other state-of-art zero-shot detectors.

1. Introduction

While deep learning based object detection methods have achieved impressive average precision over the last five years [13, 35, 32, 27, 14, 33], these gains can be attributed to the availability of training data in the form of fully annotated ground-truth object bounding boxes. Zero-Shot Detection (ZSD): The Need. As we scale up detection to large-scale applications and “in the wild” scenarios, the demand for bounding-box level annotations across a large number of object classes is simply not scalable. Consequently, as object detection moves towards large-scale 1, it is imperative that we move towards a framework that serves the dual role of detecting objects seen during training as well as detecting unseen classes as and when they appear at test-time.

1Although annotations have increased in common detection datasets (e.g. 20 classes provided by PASCAL VOC [8]; 80 in MSCOCO [26]), the size is substantially smaller relative to image classification [7].
prediction that seeks to assign higher confidence to bounding boxes that bear similarity to the synthesized unseen features as well as real seen features, while ensuring low confidence on bounding boxes that primarily contain the background. In addition, we propose a sampling scheme, whereby during training, the proposed bounding boxes are re-sampled so as to maintain a balance between background and foreground objects. Our scheme is inspired by focal loss [25], and seeks to overcome the significant foreground-background imbalance, which tends to reduce recall, and in particular adversely impacts unseen classification.

**Evaluation.** ZSD algorithms must be evaluated carefully to properly attribute gains to the different system components. For this reason we list four principal attributes that are essential for validating performance in this context:

(a) **Dataset Complexity.** Datasets such as ImageNet [7] typically contain one object/image; and F-MNIST [44] in addition has a dark background. As such, detection is somewhat straightforward obviating the need to employ DNNs. For this reason, we consider only those datasets containing multiple objects per image such as MSCOCO [26] and PascalVOC [8], where DNN detectors are required to realize high precision.

(b) **Protocol.** During training we admit images that contain only seen class objects, and filter out any image containing unseen objects (so transductive methods are omitted in our comparison). We follow [48] and consider three sets of evaluations: Test-seen (TS), Test-Unseen (TU) and Test-Mix (TM). The goal of test-seen is to benchmark performance of proposed method against vanilla detectors, which are optimal for this task. The goal of test-unseen evaluation is to evaluate performance when only unseen objects are present, which is analogous to the purely zero-shot evaluation in the recognition context [41]. Test-mix containing a mixture of both seen and unseen objects typically within the same image is the most challenging, and can be viewed analogous to generalized zero-shot setting.

(c) **High vs. Low Seen-to-Unseen Splits.** The number of objects seen during training vs. test-time determines the efficacy of the detection algorithm. At high seen/unseen object class ratios, evidently, gains are predominantly a function of recognition algorithm, necessitating no improvement object bounding boxes on unseen objects. For this reason, we experiment with a number of different splits.

(d) **AP and mAP.** Once a bounding box is placed around a valid object, the task of recognition can usually be performed by passing the bounding box through any ZSR algorithm. As such mAP performance gain could be credited to improvements in placing high-confidence bounding boxes (as reflected by AP) in the right places as well as improvements in ZSR algorithm. For instance, as we noted above, high seen/unseen ratios can be attributed to improved recognition. For this reason we tabulate APs for different splits.

2. Related Work

**Traditional vs. Generalized ZSL (GZSL).** Zero Shot Learning (ZSL) seeks to recognize novel visual categories that are unannotated in training data [21, 40, 22, 45]. As such, ZSL exhibits bias towards unseen classes, and GZSL evaluation attempts to rectify it by evaluating on both seen and unseen objects at test-time [3, 43, 12, 17, 15, 46, 37, 47]. Our evaluation for ZSD follows GZSL focusing on both seen and unseen objects.

**Generative ZSL methods.** Semantic information is a key ingredient for transferring knowledge from seen to unseen classes. This can be in the form of attributes [10, 21, 28, 29, 2, 5], word phrases [38, 11], etc. Such semantic data is often easier to collect and the premise of many ZSL methods is to substitute hard to collect visual samples for semantic data. Nevertheless, there is often a large visual-semantic gap, which results in significant performance degradation. Motivated by these concerns, recent works have proposed to synthesize unseen examples by means of generative models such as autoencoders [4, 19], GANs and adversarial methods [49, 20, 42, 16, 23, 36] that take semantic vectors as input, and output images. Following their approach, we propose to similarly bridge visual-semantic gap in ZSD through synthesizing visual features for unseen objects (since visual images are somewhat noisy).

**Zero-Shot Detection.** Recently, a few papers have begun to focus attention on zero-shot detection [1, 31, 24, 30, 48]. Unfortunately, methods, datasets, protocols and splits are all somewhat different, and the software code is not publicly available to comprehensively validate against all the methods. Nevertheless, we will highlight here some of the differences within the context of our evaluation metric (a-d).

First, [31, 24] evaluate on Test-Unseen (TU) problem. Analogous to ZSL vs. GZSL, optimizing for TU can induce an unseen class bias resulting in poor performance on seen. Furthermore, [1] tabulate GZSD performance (because purportedly mAP is low) only as a recall rate wrt top-100 bounding boxes ranked according to confidence scores. As such, there are fewer than 10 foreground objects per image, this metric is difficult to justify, since 100 bounding boxes typically includes all unseen objects. [30] proposes a transductive approach, which while evaluates seen and unseen objects, leverages appearance of unseen images during training. In contrast to these works, and like [48] we evaluate our method in the full GZSD setting.

Second, methodologically these works [1, 31, 24, 30] could be viewed as contributing to post-processing of detected bounding boxes, leveraging extensions to zero-shot recognition systems. In essence, these methods take outputs of existing vanilla detectors as given (region proposal network (RPN) [31, 30, 24] or Edge-Box [1]), and take their bounding boxes along with confidence scores as inputs into their system. This means that their gains arise primarily
from improved recognition rather than in placing bounding boxes of high-confidence. In contrast, and like [48], we attempt to improve localization performance by outputting high-confidence bounding boxes for unseen objects. Nevertheless, unlike [48], who primarily utilize semantic and visual vectors of seen classes to improve confidence of bounding boxes, we synthesize unseen visual features as well. As a result, we outperform [48] in our various evaluations.

Third, there is an issue of complexity of datasets, and as to how they are evaluated. [6] tabulates F-MNIST with clear black backgrounds, and [31] ImageNet with only single object/image, both of which are not informative from a detection perspective. Then, there is the issue of splits. A number of these methods exclusively consider high seen/unseen object class ratio splits (16/4 in [6], 177/23 in [31, 30], and 48/17 and 478/130 in [1]). Such high split ratios could be uninformative, since we maybe in a situation where the visual features of the seen class could be quite similar to the unseen class, resulting in placing sufficiently large number of bounding boxes on unseen objects. This coupled with recall@100 metric or TU evaluations could exhibit unusually high gains. Finally, AP scores are seldom tabulated, which from our viewpoint would be informative about localization performance. In contrast, and following [48] we consider several splits, different metrics (AP, mAP, recall@100) and tabulate performance on detection datasets such as MSCOCO and PascalVOC.

3. Methodology

Problem Definition. We formally define zero-shot detection (ZSD). A training dataset of $M$ images with corresponding objects labels $D_{tr} = \{I_m, \{\text{Obj}_m^{(i)}\}_{i=1}^{N_m}\}_{m=1}^M$ is provided, where $N_m$ is the number of objects and $\{\text{Obj}_m^{(i)}\}_{i=1}^{N_m}$ is the collection of all objects labels in the image $I_m$. Every object is labeled as $\text{Obj} = \{B, c\}$ where $B = \{x, y, w, h\}$ is the location and the size of the bounding box and $c \in C_{\text{seen}}$ is the class label (sup-/subscript are omitted when clear). For testing, images containing objects from both seen ($C_{\text{seen}}$) and unseen classes ($C_{\text{unseen}}$) are given with $C_{\text{seen}} \cap C_{\text{unseen}} = \emptyset$. The task is to predict the bounding box for every foreground objects. Additionally, for training the semantic representation $S_{\text{c}}$ of all classes ($c \in C_{\text{all}} = C_{\text{seen}} \cup C_{\text{unseen}}$) are also provided.

Backbone Architecture. We use YOLOv2[34] as a baseline. However, our proposed method can readily incorporate single stage detectors (SSD) or region-proposal-network (RPN). We briefly describe YOLOv2 below.

YOLOv2 is a fully convolutional network and consists of two modules: a feature extractor $F$ and an object predictor. The feature extractor is implemented by Darknet-19 [32], which takes input image size $416 \times 416$ and outputs the convolutional feature maps $F(I_m)$ with size $13 \times 13 \times 1024$. The object predictor is implemented by a $1 \times 1$ convolutional layer, which contains 5 bounding boxes predictors assigned with 5 anchor boxes with predefined aspect ratios for prediction diversity. Each bounding box predictor consists of an object locator, which outputs the bounding box location and size $\hat{B}$, a confidence predictor $\hat{C}_{\text{conf}}$, which outputs the objectness score $p_{\text{conf}}$ of the bounding box, and a classifier. The objectness score is in $[0, 1]$ and denotes the confidence of whether the bounding box contains foreground object (1) or background (0). The bounding boxes predictors convolve on every cell of $F(I_m)$ and make detection predictions for the entire image.

3.1. System Overview

The three objectives in our context are: (1) Improve bad precision-recall for unseen objects whose confidence scores are suppressed by detectors trained with seen classes; (2)
Deal with background/foreground imbalance that hampers the precision; and (3) Account for generalized ZSD performance where both seen/unseen objects exist in the test set.

**Key Idea.** All of these objectives can be realized by improving the confidence predictor component, whereby both seen and unseen object bounding boxes receive higher confidence while background objects are still suppressed. To do so we retrain confidence predictor by leveraging real visual features for seen and background objects, and synthetic features for unseen objects. We resample bounding boxes to correct the background/foreground imbalance. Fig. 1 depicts the four stages of the proposed pipeline:

1. **Pre-training.** Extract confidence predictor component after training a stand-alone detector on training data.
2. **Re-sampling.** Re-sample foreground (seen objects) and background bounding boxes in the training set so that they are equally populated;
3. **Visual Feature Generation.** Train generator using the visual features of bounding boxes in (2) and semantic data to synthesize visual features for unseen classes;
4. **Confidence Predictor Re-training** Retrain confidence predictor with the real and synthetic visual features, and plug it back into the original detector.

Following [34] we train YOLOv2 for Step 1. We describe the other steps in the sequel.

### 3.2. Foreground/Background Re-Sampling

Our objective is to construct a collection of visual features of (seen) foreground objects and background objects from the training set to reflect a balanced ratio of foreground/background objects. Note that the cell convolutional feature $F(I_m)$ is an inexpensive but effective visual representation of the bounding box proposals predicted by the current cell. However, not all cells are suitable to represent a bounding box since a cell may only overlap with the objects partially thus not sufficiently representative of the desired bounding box. Therefore, the re-sampled visual feature set $D_{r es}^{\alpha, \beta}$ (where $\alpha$ refers to the cell location and $\beta$ the bounding box index) is constructed as follows:

**Foreground.** For every image $I_m$, a cell feature $f$ is viewed as foreground if its associated bounding box prediction $B$ has a maximum Intersection over Union (IoU) greater than 0.5 on ground truth objects collections $\{Obj_{m_i}^{(j)}\}_{i=1}^{N_m}$, as well as its confidence prediction $\hat{p}_{\text{conf}}$ is higher than 0.6. The feature $f$, along with its confidence score $\hat{p}_{\text{conf}}$ and the ground truth class label $c$ of the object with maximum IoU, will be treated as a data point $(f, \hat{p}_{\text{conf}}, c)$.

**Background.** A cell feature is viewed as background if its maximum IoUs over all ground truth objects is smaller than 0.2 and the associated confidence score is lower than 0.2.

Features with top $r \times K_m$ smallest maximum IoUs will be selected as the background data points $(f, \hat{p}_{\text{conf}}, c_{bg})$, where $K_m$ is the number of foreground features extracted on image $I_m$, $r$ is the ratio of foreground/background data, and $c_{bg}$ is the class label for background which we set to -1.

In our experiments, we let $r = 1$ to balance the background and foreground objects.

In the sequel to avoid clutter, we omit the superscripts in $D_{r es}$ and write the re-sampled visual feature set as $D_{r es}$.

### 3.3. Visual Feature Generation

After $D_{r es}$ is constructed, the next step is to train a visual feature generator to synthesize those features from their semantic counterparts while minimizing information loss. In particular, we construct our generator based on the concept of a conditional variational auto-encoder (CVAE) [39], but add an additional visual consistency checker component $D$ to provide more supervision, as shown in Fig. 2. The CVAE is conditioned on the class semantic representation $S_c$, consisting of an encoder $E$ and a decoder $G$. The encoder takes the input as a concatenation of seen feature $f$ and semantic attribute $S_c$, and outputs the distribution of the latent variable $z$: $p_E(z|f, S_c)$. The decoder then generates exemplar feature $\hat{f}$ given the latent random variable $z$ and class semantic $S_c$: $\hat{f} = G(z, S_c)$. The visual consistency checker $D$ provides three additional supervisions on the generated feature $\hat{f}$ in addition to the reconstruction loss for CVAE: confidence consistency, attribute consistency loss, and classification consistency as described below.

**Conditional VAE.** The decoder $G$ with parameters $\theta_G$, is responsible for generating unseen exemplar features which will be further used to retrain the confidence predictor. $\theta_G$ is trained along with $\theta_E$, the parameters of the encoder, by the conditional VAE loss function as following:

$$\ell_{\text{CVAE}}(\theta_G, \theta_E) = \KL(p_E(z|f, S_c)||p(z)) - \mathbb{E}_{p_G}(\log p_G(f|z, S_c))$$

(1)

where the first term on right hand side is the KL divergence between the encoder posterior and prior of latent variable $z$, and the second term is the reconstruction error. Minimizing the KL divergence will enforce the conditional posterior distribution approximates the true prior. Following [18], we utilize an isotropic multivariate Gaussian and the reparameterization trick to simplify these computations.
**Visual Consistency Checker.** This component provides multiple supervisions to encourage the generated visual features $\hat{f}$ to be consistent with the original feature $f$:

**Confidence Consistency:** The reconstructed feature $\hat{f}$ should have the same confidence score as the original one, therefore, the confidence consistency loss is defined as the mean square error (MSE) between the confidence score of reconstructed and original features:

$$
\ell_{\text{conf}}(\theta_G) = \mathbb{E}_{D_{\text{res}}} \left[ \hat{p}_{\text{conf}} - \text{Conf}(\hat{f}) \right]^2
$$

where $\text{Conf}(\cdot)$ refers to the confidence predictor model whose weights is frozen here for training the visual feature generator.

**Classification Consistency:** The reconstructed feature $\hat{f}$ should also be discriminating enough to be recognized as the original category. Therefore, we feed $\hat{f}$ in to the classifier $\text{Clf}$ and penalize the generator with the cross-entropy loss:

$$
\ell_{\text{clf}}(\theta_G) = \mathbb{E}_{D_{\text{res}}} \left[ \text{CE}(\text{Clf}(\hat{f}), c) \right]
$$

where $c \in C_{\text{seen}} \cup \{-1\}$ is the ground truth class for $f$, and $\text{Clf}$ is pretrained by cross-entropy loss on $D_{\text{res}}$ and will not be updated when training the generator. A class-weighted cross-entropy loss can also be used here to balance the data. 

**Attribute Consistency:** The generated feature should also be coherent with its class semantic. We thus add an attribute consistency loss $\ell_{\text{attr}}$ which back-propagates error to the generator between the attribute predicted on $\hat{f}$ and the conditioned class semantic:

$$
\ell_{\text{attr}}(\theta_G) = \mathbb{E}_{D_{\text{res}}} \left[ S_c - \text{Attr}(\hat{f}) \right]^2
$$

where $S_{-1} = 0$ zero vector for background. The predictor $\text{Attr}$ is also pretrained on $D_{\text{res}}$ and the weights are frozen when optimizing $\ell_{\text{attr}}$. Different class weights can also be applied for the purpose of data balance because the number of background is much larger than the other classes.

The parameters of CVAE can be end-to-end learned by minimizing the weighted sum of the CVAE and visual consistency checker loss functions:

$$
\theta^*_G, \theta^*_E = \arg \min_{\theta_G, \theta_E} \ell_{\text{CVAE}} + \lambda_{\text{conf}} \ell_{\text{conf}} + \lambda_{\text{clf}} \ell_{\text{clf}} + \lambda_{\text{attr}} \ell_{\text{attr}}
$$

where $\lambda_{\cdot, \cdot}$ are the weights for the respective loss terms.

After the CVAE is trained, we can synthesize data feature for both seen and unseen objects by feeding the corresponding class attribute $S_c$ and latent variable $z$ randomly sampled from the prior distribution $p(z)$ to the decoder $G$. We generate $N_{\text{seen}}$ examples for every seen class and $N_{\text{unseen}}$ for every unseen class. We assume every synthesized data is ground truth object and assign 1 as its target confidence score, thus the synthesized data is constructed as $D_{\text{syn}} = \{\hat{f}, 1, c\}$ where $c \in C_{\text{all}}$.

### 3.4. Confidence Predictor Re-Training

With the collected real features $D_{\text{res}}$ and the synthetic visual features $D_{\text{syn}}$ which contains the generated visual features of unseen classes, we are now ready to re-train the confidence predictor $\text{Conf}(\cdot)$ to encourage its confidence prediction on unseen objects while retaining its performance on seen and background objects. Specifically, $\text{Conf}(\cdot)$ is then re-trained on the combination of the extracted and synthetic features $D_{\text{res}} \cup D_{\text{syn}}$. Following the original YOLOv2\[34], MSE loss is used and the formal loss function is defined as:

$$
\ell = \frac{1}{|D_{\text{res}}|} \sum_{f \in D_{\text{res}}} |\text{Conf}(f) - \hat{p}_{\text{conf}}|^2 + \frac{1}{|D_{\text{syn}}|} \sum_{f \in D_{\text{syn}}} |\text{Conf}(\hat{f}) - 1|^2
$$

We load weights from pretrained YOLOv2 in the first step (see Sec 3.1) as a warm-start. Training with loss, $\ell$, encourages the confidence predictor to output higher scores for unseen object features while preserving existing confidences for seen objects and background.

### 3.5. Implementation Details

The encoder $E$ and decoder $G$ are both two fc-layer networks in our CVAE model. The input size of $E$ is $N_{\text{feat}} + N_{\text{attr}}$ where $N_{\text{feat}} = 1024$ is the feature size and $N_{\text{attr}}$ is the length of the class semantic. The output size of $E$ which is also the size of latent variable $z$, $N_{\text{latent}}$, is set to 50. The input size of $G$ is $N_{\text{latent}} + N_{\text{attr}}$ and the hidden layer of $E$ and $G$ has 128 nodes.

For the visual consistency checker, both the classifier $\text{Clf}(\cdot)$ and attribute predictor $\text{Attr}(\cdot)$ are paramerized by a two FC-layer networks with hidden size 256. When pretrained on $D_{\text{res}}$, $\text{Clf}(\cdot)$ is trained 5 epochs with learning rate $1e-4$ and $\text{Attr}(\cdot)$ is trained 10 epochs with learning rate $1e-4$, respectively. We set $\lambda_{\text{conf}} = 1$, $\lambda_{\text{clf}} = 2$ and $\lambda_{\text{attr}} = 1$. We generate $N_{\text{seen}} = 50$ examples for every seen classes and $N_{\text{unseen}} = 1000$ for unseen.

### 4. Experiments

To evaluate the performance of our method, DELO, we conduct extensive qualitative and quantitative experiments. We tabulate results against other recent state-of-the-art methods, and then perform ablative analysis to identify important components of our system. We follow the protocol of \[48], which emphasizes the need for evaluation both seen and unseen examples at test-time. As in \[48] we consider only visual seen examples during training. In summary, (1) Consider generalized ZSD setting and omit results for the transductive generalized setting of \[30] and somewhat de-emphasize the purely unseen detection results of \[1, 31, 24]; (2) Consider multiple splits with various
Figure 3. Visual examples of our ZSD results. Each triple shows: (from left to right) DELO detection results, vanilla YOLOv2 detection results at the same confidence threshold as DELO, vanilla YOLOv2 detection results at a much smaller confidence threshold. The seen, unseen and errors are color-coded as red, green and blue. Notice that compared to DELO, the vanilla YOLOv2 constantly predicts extremely low objectness scores on unseen objects, and suffers from significant detection errors for those unseen objects to be detected.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pascal VOC split</th>
<th>MS COCO split</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TU</td>
<td>TS</td>
<td>TM</td>
</tr>
<tr>
<td>YOLOv2</td>
<td>5/15</td>
<td>36.6</td>
<td>85.6</td>
</tr>
<tr>
<td>ZS-YOLO</td>
<td>37.3</td>
<td>85.0</td>
<td>30.9</td>
</tr>
<tr>
<td>DELO</td>
<td>39.4</td>
<td>88.2</td>
<td>34.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Pascal VOC split</th>
<th>MS COCO split</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TU</td>
<td>TS</td>
<td>TM</td>
</tr>
<tr>
<td>YOLOv2</td>
<td>10/10</td>
<td>56.4</td>
<td>71.6</td>
</tr>
<tr>
<td>ZS-YOLO</td>
<td>60.1</td>
<td>71.0</td>
<td>53.9</td>
</tr>
<tr>
<td>DELO</td>
<td>61.3</td>
<td>73.5</td>
<td>59.6</td>
</tr>
</tbody>
</table>

Table 1. Zero-shot detection evaluation results on various datasets and 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 for every setting is in **bold**.

<table>
<thead>
<tr>
<th>Method</th>
<th>TU</th>
<th>TS</th>
<th>TM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv2</td>
<td>56.4</td>
<td>71.6</td>
<td>54.3</td>
<td></td>
</tr>
<tr>
<td>BS-1</td>
<td>59.5</td>
<td>73.2</td>
<td>58.5</td>
<td></td>
</tr>
<tr>
<td>BS-2</td>
<td>60.6</td>
<td>73.4</td>
<td>59.0</td>
<td></td>
</tr>
<tr>
<td>BS-3</td>
<td>61.0</td>
<td>73.4</td>
<td>59.4</td>
<td></td>
</tr>
<tr>
<td>DELO</td>
<td>61.3</td>
<td>73.5</td>
<td>59.6</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Evaluation on the 10/10 split of Pascal VOC for baseline models. TU = Test-Unseen, TS = Test-Seen, TM = Test-Mix. Overall average precision in % is reported. The difference between original YOLOv2 is reported in (·) and the highest difference is in **bold**.

seen/unseen ratios in contrast to tabulating results for single splits with large seen/unseen ratios by [31, 1]); (3) consider multi-object image datasets, and results for other datasets such as F-MNIST that has clear black backgrounds as in [6] or single objects/image such as ImageNet as in [31, 30]). More detailed discussions can be found in Sec. 1 and Sec. 2.

Datasets. We choose Pascal VOC [9] and MSCOCO [26], both of which are well known detection benchmarks, and as such exhibit multiple objects per image. PascalVOC has only 20 classes. For this reason, our goal here is to primarily understand how performance varies with different split ratios of seen to unseen objects (5/15, 10/10, and 15/5). MSCOCO is a larger dataset with about 80 classes and serves the purpose of understanding performance for fixed collection of unseen classes as the number of seen classes increase (20, 40 to 60).

**Setting.** For each seen/unseen split, we evaluate our method on three data configurations: Test-Seen (TS), Test-Unseen
(TU), and Test-Mix (TM) [48]. For Test-seen our test images only contain objects from seen classes; test-unseen are images that only contain unseen objects; and test-mix are those that contain both seen and unseen objects. Test-mix is the generalized ZSD setting and is the most challenging, where the model needs to detect seen and unseen objects simultaneously. Following [48], we also use 0.5-IoU and 11 points average precision (AP) for evaluation.

**Semantic Information.** Following [48], we use the attribute annotation from aPY [10] as the semantics on Pascal VOC. The semantic vectors are obtained by averaging the object-level attribute of all examples in the class. We use PCA to reduce dimensions to 20 to mitigate noise. On MSCOCO, a 25-dim word embedding w2vR proposed in [48] is used. The semantics are re-scaled to [0, 1] on each dimension for both Pascal and MSCOCO.

**Training Details.** For Pascal VOC, CVAE is trained by an Adam optimizer with a learning rate of $\text{1e}^{-4}$. On the 10/10 and 15/5 splits, we set training epochs to 60 and scale the learning rate by 0.5 every 15 epochs. On the 5/15 split, the training epoch is 200 and the learning rate is scaled by 0.5 every 60 epochs. On MSCOCO, the learning rate is set to $1e^{-4}$. On 20/20 split the model is trained for 60 epochs, while on 40/20 and 60/20 splits, the model is trained for 40 epochs. The learning rate is scaled by 0.5 every 15 epochs.

## 4.1. Zero Shot Detection Evaluation

**Tabulating AP.** We evaluate DELO on all seen/unseen splits as well as Test-Seen/Unseen/Mix configurations (Table 1) against vanilla YOLOv2 [34] trained in a standard fully-supervised manner with the training partition, as well as the state-of-the-art ZSD method ZS-YOLO [48].

**Discussion Part-A.**

(1) **Vanilla YOLOv2 does well on Test-Seen.** The state-of-the-art YOLOv2 as reported in [34] are 73.4% mAP on Pascal VOC2012 and 44.0% mAP on MSCOCO. Observe from Table 1 that the vanilla YOLOv2 trained on seen partition achieves similar performances on Test-Seen with no unseen objects, i.e. 85.6%, 71.6%, 75.3% for Pascal VOC (5/15, 10/10, 15/5 split, respectively), and 34.5%, 48.7%, 44.8% for MS COCO (20/20, 40/20, 60/20 split, respectively). Consequently, YOLOv2 is a strong baseline to compare against particularly for test-seen. Furthermore, as we increase the split ratio the number of seen classes increases, and consequently, test-mix tends to favor seen class detection. For this reason we should expect vanilla YOLOv2 detector to perform better in this case as well.

(2) **Re-training with synthetic visual features improves detection performance.** DELO consistently outperforms vanilla YOLOv2 and ZS-YOLO on all test configurations. ZS-YOLO uses semantic features to train the confidence predictor, which can be visually noisy (attributes such as “useful”). As a result, while improving upon YOLOv2 on Test-Unseen/-Mix, its Test-seen, performance is lower, e.g. MS COCO 40/20 split, ZS-YOLO gets 44.0% on Test-Seen, compared to YOLOv2’s 48.7%. In contrast, DELO’s confidence predictor leverages visual features from seen/unseen/background boxes. Additionally, the feature pool is re-sampled according to a more balanced foreground/background ratio. Consequently, DELO, also improves Test-Seen performance, e.g. MS COCO all splits, we see average DELO’s AP is (2.53% / 8.63% / 11.53%) and (7.25% / 7.90% / 14.63%) better than ZS-YOLO and YOLOv2 respectively, on Test-Unseen/Seen/Mix.

(3) **DELO is robust to different seen/unseen configurations.** YOLOv2 and ZS-YOLO’s performance changes significantly with large number of classes (MSCOCO). As seen classes increases and unseen classes remain the same, YOLOv2 realizes (12.3% / 24.6% / 37.6%) on Test-Mix; ZS-YOLO realizes (20.2% / 30.0% / 33.6%). Compared to these, DELO produces a much more consistent detection performance (41.6% / 37.5% / 39.4%). On Pascal VOC, performance of all the three models varies significantly for different splits of Test-Mix since the dataset is of a smaller scale and the number of unseen classes is also changing. But DELO’s performance is still superior.

**Tabulating Recall@100 and mAP.** Following the protocol in [1], we conducted a second set of experiments on MS COCO, adopting the Recall@100 and mAP as evaluation metrics, to baseline against [1, 24] (more details in Sec. 2). The configurations and ZSD performance are reported in Table 3.

**Discussion Part-B.**

ZSD is in essence a classification problem under the Recall@100 metric at high seen/unseen ratio. Observe that, a vanilla detector, e.g. YOLOv2 in cascade with a ZSR model (we chose a 2-FC classifier trained on $D_{syn}$) achieves similar performance on Recall@100 as existing ZSD methods, i.e. 24.8 (YOLOv2) compared to 24.4 (SB [1]), 27.2 (DSES [1]), 34.3 ($\text{DELO}$ [24]). Fundamentally, the issue is two-fold. First at large split ratio’s the current methods benefit from unseen visual features that resemble seen exam-

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@100</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZSD</td>
<td>GZSD</td>
</tr>
<tr>
<td>SB[1]</td>
<td>24.4</td>
<td>-</td>
</tr>
<tr>
<td>DSES[1]</td>
<td>27.2</td>
<td>15.2</td>
</tr>
<tr>
<td>TD[24]</td>
<td>34.3</td>
<td>-</td>
</tr>
<tr>
<td>YOLOv2</td>
<td>24.8 (51.6)</td>
<td>30.8 (52.8)</td>
</tr>
<tr>
<td>DELO</td>
<td>33.5 (55.7)</td>
<td>36.6 (55.8)</td>
</tr>
</tbody>
</table>

Table 3. ZSD and GZSD performance evaluated with Recall@100 and mAP on MS COCO to compare with other ZSD methods. A 2-FC classifier trained on $D_{syn}$ is appended to YOLOv2 and DELO to conduct the full detection. The number in the parenthesis is class-agnostic recall ignoring classification.
amples, and so do not require better detections. Vanilla detectors that are not optimized for unseen objects are capable of localizing unseen objects. Second, the Recall@100 metric also helps in this process since 100 bounding boxes typically contain all unseen objects at the high split ratios. Once this is guaranteed, background boxes can be eliminated based on post-processing with a zero-shot classifier that rejects background whenever no unseen class is deemed favorable. For this reason, we also present other metrics such as AP in Table 1 as well as (in Table 3 within brackets) whether bounding boxes are true objects. In addition we see that mAP improves both under ZSD as well as the more important GZSD setting. Finally, while TD is marginally better on ZSD, we emphasize that it is possible to bias ZSR models towards unseen classes when we are cognizant of the fact that no seen classes are present [41]. Note that a large fraction of our bounding boxes are indeed correct, and so our lower performance can be attributed to the fact that we did not fine tune our ZSR model.

4.2. Ablative Analysis

Contribution of visual consistency checker. The visual consistency checker $D$ in our generative model provides more supervision to the decoder to encourage it to generate better exemplar features. To measure the contribution of each components in the visual consistency checker, we compare with three baselines: (1) BS-1: the entire visual consistency checker $D$ is removed, the model is thus reduced to a standard CVAE and trained only by $\ell_{\text{CVAE}}$. (2) BS-2: only the confidence predictor is used in the visual consistency checker and the model is trained by $\ell_{\text{CVAE}} + \lambda_{\text{conf}} \cdot \ell_{\text{conf}}$. (3) BS-3: the attribute predictor is removed from the visual consistency checker and the model is trained by $\ell_{\text{CVAE}} + \lambda_{\text{conf}} \cdot \ell_{\text{conf}} + \lambda_{\text{clff}} \cdot \ell_{\text{clff}}$. We evaluate the baseline models on 10/10 split of Pascal VOC and the results along with the differences between the original YOLOv2 are tabulated in Table 2.

It is apparent that with all the visual consistency checker components included, DELO realizes optimal performance. Without any supervision from the consistency checker, the pure CVAE achieves 59.5 on TU, 73.2 on TS and 58.5 on TM (BS-1). Incorporating the confidence predictor $\text{Conf}$ increases 1.1 on TU and 0.5 on TM, and the classifier $\text{Clff}$ contributes 0.3 improvement on TU and 0.4 on TM. Finally, by integrating the attribute predictor, the performance further increases 0.3 on TU, 0.1 on TS, and 0.2 on TM. The visual consistency checker improves the overall performance, especially on TU and TM, as it encourages the generated data to be more consistent to the original feature and the class information.

Number of Generated Examples. We also perform experiments to evaluate how the number of generated examples affects the detection performance. In the experiment, we first vary $N_{\text{seen}}$ in the range $[20, 50, 100, 200, 500]$ while keeping $N_{\text{unseen}} = 1000$. Then we vary $N_{\text{unseen}}$ in the range of $[0, 100, 200, 500, 1000, 2000]$ while set $N_{\text{seen}} = 50$. The experiments are conducted on 10/10 split of Pascal VOC and the final detection performance are visualized in Fig. 4. The generated unseen data plays an important role in the method, as we can see the performance on TU and TM drops $>2\%$ when training with $N_{\text{unseen}} = 0$. The performance on TU and TM increases when more unseen data are available, and get saturated after $N_{\text{unseen}} > 1000$. A small number of unseen examples (e.g. 100) is sufficient for learning a strong confidence predictor. The number of seen generated data, on the other hand, only affects the overall performance slightly as it has similar distribution as $D_{res}$.

5. Conclusion

We proposed DELO, a novel Zero-shot detection algorithm for localizing seen and unseen objects. We focus on the generalized ZSD problem where both seen and unseen objects can be present at test-time, but we are only provided examples of seen objects during training. Our key insight is that, while vanilla DNN detectors are capable of producing bounding boxes on unseen objects, these get filtered out due to poor confidence. To address this issue DELO synthesizes unseen class visual features, leveraging semantic data. Then a confidence predictor is trained with training data augmented with synthetic features. We employ a conditional variational encoder, with additional loss functions, that are specifically chosen to improve detection performance. We also propose a re-sampling strategy to improve the foreground/background during training. Our results show that on a number metrics, on complex datasets involving multiple objects/image, DELO achieves state-of-the-art performance.

Acknowledgement

This work was supported partly by the National Science Foundation Grant 1527618, the Office of Naval Research Grant N0014-18-1-2257 and by a gift from ARM corporation.
References


Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE transactions on pattern analysis and machine intelligence, 2018.


