Transferable Semi-Supervised 3D Object Detection From RGB-D Data

Yew Siang Tang  
Gim Hee Lee
Department of Computer Science, National University of Singapore
{yewsiang.tang, gimhee.lee}@comp.nus.edu.sg

Abstract

We investigate the direction of training a 3D object detector for new object classes from only 2D bounding box labels of these new classes, while simultaneously transferring information from 3D bounding box labels of the existing classes. To this end, we propose a transferable semi-supervised 3D object detection model that learns a 3D object detector network from training data with two disjoint sets of object classes - a set of strong classes with both 2D and 3D box labels, and another set of weak classes with only 2D box labels. In particular, we suggest a relaxed reprojection loss, box prior loss and a Box-to-Point Cloud Fit network that allow us to effectively transfer useful 3D information from the strong classes to the weak classes during training, and consequently, enable the network to detect 3D objects in the weak classes during inference. Experimental results show that our proposed algorithm outperforms baseline approaches and achieves promising results compared to fully-supervised approaches on the SUN-RGBD and KITTI datasets. Furthermore, we show that our Box-to-Point Cloud Fit network improves performances of the fully-supervised approaches on both datasets.

1. Introduction

3D object detection refers to the problem of classification and estimation of a tight oriented 3D bounding box for each object in a 3D scene represented by a point cloud. It plays an essential role in many real-world applications such as home robotics, augmented reality and autonomous driving, where machines must actively interact and engage with its surroundings. In the recent years, significant advances in 3D object detection have been achieved with fully supervised deep learning-based approaches [4, 5, 6, 12, 16, 20, 31, 35, 40]. However, these strongly supervised approaches rely on 3D ground truth datasets [7, 25, 32, 33] that are tedious and time-consuming to label even with good customised 3D annotation tools such as those in [7, 25]. In contrast, calculations based on [17] show that it can be ~3-16 times faster (depending on the annotation tool used) to label 2D than 3D bounding boxes (details in supplementary). This observation inspires us to investigate the direction of training a 3D object detector for new object classes from only 2D bounding box labels of these new classes, while simultaneously transferring information from 3D bounding box labels of the existing classes. We hope that this direction of research leads to cost reductions, and improves the efficiency and practicality of learning 3D object detectors in new applications with new classes.

More specifically, the objective of this paper is to train a 3D object detector network with data from two disjoint sets of object classes - a set of classes with both 2D and 3D box labels (strong classes), and another set of classes with only 2D box labels (weak classes). In other words, our goal is to transfer useful 3D information from the strong classes to the weak classes (the new classes of a new 3D object detection application correspond to these weak classes). We refer to this as cross-category semi-supervised 3D object detection as illustrated in Fig. 1.

Figure 1. Illustration of fully-supervised 3D object detection where 3D box labels of all classes (weak and strong classes) are available, and cross-category semi-supervised 3D object detection where 3D box labels of inference classes (weak classes e.g. Table and Night stand) are not available. Conventional semi-supervised learning (which is in-category) requires strong annotations from the inference classes (3D box labels in the case of 3D object detection) while cross-category semi-supervised learning requires only the weak annotations from the inference classes (2D box labels in the case of 3D object detection).

Table 1. The 3 sets of object classes - a set of strong classes with both 2D and 3D box labels, and another set of weak classes with only 2D box labels (weak classes).
To this end, we propose a novel transferable semi-supervised 3D object detection network. Our network leverages on the state-of-the-art Frustum PointNets \cite{PoinTrunc2018} supervised 3D object detection network. Our network X to mask X takes the 3D box predictions on the method to encode a 3D box and point cloud pair. Finally, and the fit with their corresponding point clouds. More 3D box and point cloud pairs to reason about 3D boxes first train a Box-to-Point Cloud (BoxPC) Fit network on initial 3D box predictions on the \cite{PoinTrunc2018} box prior losses on the \cite{PoinTrunc2018} classes. We also supervise the network with 2D box labels of the \cite{PoinTrunc2018} classes using a relaxed reprojection loss which corrects 3D box predictions that violate boundaries specified by the 2D box labels and through the prior knowledge of the object sizes. To transfer knowledge from 3D box labels from the \cite{PoinTrunc2018} classes, we first train a Box-to-Point Cloud (BoxPC) Fit network on 3D box and point cloud pairs to reason about 3D boxes and the fit with their corresponding point clouds. More specifically, we proposed an effective and differentiable method to encode a 3D box and point cloud pair. Finally, the 3D box predictions on the \cite{PoinTrunc2018} classes are supervised and refined by the BoxPC Fit network.

The contributions of our paper are as follows:

- We propose a network to perform 3D object detection on \cite{PoinTrunc2018} classes, where only 2D box labels are available. This is achieved using relaxed reprojection and box prior losses on the \cite{PoinTrunc2018} classes, and transferred knowledge from the \cite{PoinTrunc2018} classes.

- A differentiable BoxPC Fit network is designed to effectively combine BoxPC representation. It is able to supervise and improve the 3D object detector on the \cite{PoinTrunc2018} classes after training on the \cite{PoinTrunc2018} classes.

- Our transferable semi-supervised 3D object detection model outperforms baseline approaches and achieves promising results compared to fully-supervised methods on both SUN-RGBD and KITTI datasets. Additionally, we show that our BoxPC Fit network can be used to improve the performance of fully-supervised 3D object detectors.

2. Related Work

3D Object Detection 3D object detection approaches have advanced significantly in the recent years \cite{PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018, PoinTrunc2018}. However, most approaches are fully-supervised and highly-dependent on 3D box labels that are difficult to obtain. To the best of our knowledge, there are no existing weakly- or semi-supervised 3D object detection approaches. Recently, \cite{PoinTrunc2018} proposed a self-supervised Augmented Autoencoder that trains on CAD models, removing the need for pose-annotated training data. However, it is unclear if the method generalizes to general 3D object detection datasets with high intra-class shape variations and noisy depth data.

supervised [8, 15, 18, 36] learning approaches for semantic segmentation. Both strong and weak labels on the inference classes are required in conventional semi-supervised learning. Hence, the approach remains expensive for applications with new classes. [9, 10, 34] proposed the cross-category semi-supervised semantic segmentation which is a more general approach of semi-supervised segmentation, where the model is able to learn from strong labels provided for classes outside of the inference classes. They outperformed weakly- and semi-supervised methods, and showed on-par performances with fully-supervised methods. Inspired by the effectiveness of the transferred knowledge, we propose to tackle the same problem in the 3D object detection domain.

3. Problem Formulation

Let \( C = C_{2D} \cup C_{3D} \), where \( C_{2D} \) is the set of classes with only 2D box labels (weak classes) and \( C_{3D} \) is the disjoint set of classes with 2D and 3D box labels (strong classes). We tackle the cross-category semi-supervised 3D object detection problem where 3D object detection is performed on the object classes in \( C_{2D} \) while training is done on strong labels from \( C_{3D} \) and weak labels from \( C_{2D} \). In 3D object detection with RGB-D data, our goal is to classify and predict amodal 3D bounding boxes for objects in the 3D space. Depth data can be provided by various depth sensors e.g. RGB-D sensors and LiDAR etc. Each 3D bounding box is parameterized by its center \((b_x, b_y, b_z)\), size \((b_l, b_w, b_h)\) and orientation \(b_\theta\) along the vertical axis.

4. Method

4.1. Frustum PointNets

In this section, we briefly discuss Frustum PointNets (FPN), the 3D object detection framework which we adopt as the backbone of our framework. Refer to [20] and our supplementary for more details.

4.1.1 Frustum Proposal

As seen in Fig. 2, the inputs to the network is an image with a 2D box for each object and a 3D point cloud. During inference, we obtain the 2D boxes from a 2D object detector trained on \( C \). We project the point cloud onto the image plane using the camera projection matrix and select only the points that lie in the 2D box for each object. This reduces the search space to only points within a 3D frustum which we refer to as frustum point cloud. Next, variance of the points in each frustum point cloud is reduced by rotating the frustum about the vertical axis of the camera to face the front. Formally, let us denote a frustum point cloud by \( X = [x_1, x_2, \ldots, x_N]^T \in \mathbb{R}^{N \times (3+k)} \), where \( N \) is the number of points. \( x_n \in \mathbb{R}^{3+k} \) (i.e., \( x, y, z \), RGB or reflectance) is the \( n \)-th point in the point cloud.

4.1.2 3D Instance Segmentation

The frustum point cloud contains foreground points of an object which we are interested in and background points from the surroundings. To isolate the foreground object and simplify the subsequent 3D box estimation task, the frustum point cloud is fed into a 3D instance segmentation network \( f_{seg} \) which predicts a class-agnostic foreground probability for each point. These probabilities are used to mask and retrieve the points with high values, thus giving a masked point cloud. \( f_{seg} \) is trained by minimizing

\[
L_{seg}^{3D} = \sum_{c \in C_{3D}} H(f_{seg}(X^c), \hat{f}_{seg}(X^c)) \tag{1}
\]

on \( C_{3D} \), where \( H \) is the point-wise binary cross-entropy loss and \( \hat{f}_{seg} \) is the ground truth mask.

4.1.3 3D Box Estimation

The 3D box estimation network \( f_{box} \) predicts an initial 3D box of the object \( B_0 \) given the masked point cloud and a one hot class vector from the 2D detector. Specifically, we compute the centroid of the masked point cloud and use it to translate the masked point cloud to the origin position to reduce translational variance. Next, the translated point cloud is given to two PointNet-based [21] networks that will predict the initial 3D box \( B_0 \). The center \((b_x, b_y, b_z)\) is directly regressed by the network while an anchor-based approach is used to predict the size \((b_l, b_w, b_h)\) and rotation \(b_\theta\). There are \( NS \) size and \( NH \) rotation anchors, and \( 3NS \) size and \( NH \) rotation residuals. Hence, the network has \( 3 + 4NS + 2NH \) outputs that are trained by minimizing

\[
L_{box}^{3D} = \sum_{c \in C_{3D}} L_{c1-reg}^c + L_{c2-reg}^c + L_{r-cla}^c + L_{r-reg}^c
\]

\[
+ L_{s-cla}^s + L_{s-reg}^s + L_{corner}^c \tag{2}
\]

on \( C_{3D} \). \( L_{c1-reg}^c \) and \( L_{c2-reg}^c \) are regression losses for the box center predictions, \( L_{r-cla}^c \) and \( L_{r-reg}^c \) are the rotation anchor classification and regression losses, \( L_{s-cla}^s \) and \( L_{s-reg}^s \) are the size anchor classification and regression losses, and \( L_{corner}^c \) is the regression loss for the 8 vertices of the 3D box. Note that each loss term in \( L_{box}^{3D} \) is weighted...
by $w_s$, which we omit in Eq. 2 and all subsequent losses for brevity. Until this point, the network is fully-supervised by 3D box labels available only for classes $C_{3D}$. In the subsequent sections, we describe our contributions to train $f_{box}$ on the weak classes in $C_{2D}$ by leveraging on the BoxPC Fit Network (Sec. 4.2) and weak losses (Sec. 4.3).

### 4.2. Box to Point Cloud (BoxPC) Fit Network

The initial 3D box predictions $\hat{B}_0$ on classes $C_{2D}$ are not reliable because there are no 3D box labels for these classes. As a result, the 3D box surfaces of the initial predictions $\hat{B}_0$ for classes $C_{2D}$ are likely to cut the frustum point cloud $X$ at unnatural places, as illustrated on the right of Fig. 3. We utilize our BoxPC Fit Network $f_{boxpc}$ with a novel training method to transfer the knowledge of a good BoxPC fit from the 3D box labels of classes $C_{3D}$ to classes $C_{2D}$. The input to our $f_{boxpc}$ is a pair of 3D box $B$ and frustum point cloud $X$. The outputs of $f_{boxpc}$ are the BoxPC Fit probability, i.e. goodness-of-fit

$$\hat{p} = f_{boxpc-cls}(X, B) \in [0, 1],$$

(3)

between $B$ and the object in $X$, and the correction

$$\Delta \hat{B} = f_{boxpc-reg}(X, B) \in \mathbb{R}^7,$$

(4)

required on $B$ to improve the fit between $B$ and the object in $X$, $\hat{p} \rightarrow 1$ when $B$ encloses the object in $X$ tightly, i.e. there is a high overlap between $B$ and the 3D box label $B^*$. A pretrained $f_{boxpc}$ (see Sec. 4.2.1 for the training procedure) is used to supervise and improve 3D box predictions $\hat{B}$ for classes $C_{2D}$. Specifically, we train $f_{box}$ to make better initial 3D box predictions $\hat{B}_0$ by maximizing $\hat{p}$. By minimizing the loss:

$$L^{2D}_{fit} = \sum_{c \in C_{2D}} -\log(f_{boxpc-cls}(X^c, \hat{B}_0^c)),$$

(5)

i.e. maximizing $\hat{p}$, our network $f_{box}$ learns to predict $\hat{B}_0$ that fit objects in their respective point clouds well. Finally, we obtain the final 3D box prediction $\hat{B}$ by correcting the initial 3D box prediction $\hat{B}_0$ with $\Delta \hat{B}$, i.e. $\hat{B} = \hat{B}_0 + \Delta \hat{B}$.

#### 4.2.1 Pretraining BoxPC Fit Network

We train the BoxPC Fit network $f_{boxpc}$ on the classes in $C_{3D}$ with the 3D box labels $B^* \in \mathbb{R}^7$ and their corresponding point clouds $X$. We sample varying degree of perturbations $\delta \in \mathbb{R}^7$ to the 3D box labels and get perturbed 3D boxes $B^* - \delta$. We define 2 sets of perturbations, $P^+(B^*)$ and $P^-(B^*)$, where $P^+$ and $P^-$ are sets of small and large perturbations, respectively. Formally, $P^+(B^*) = \{\delta : \alpha^+ \leq IOU(B^* - \delta, B^*) \leq \beta^+ | \{\alpha^+, \beta^+\} \in [0, 1]\}$, where $IOU$ refers to the 3D Intersection-over-Union. $P^-$ is defined similarly but with $\alpha^-$ and $\beta^-$ as the bounds. $\alpha^+$ and $\beta^+$ are set such that the 3D boxes perturbed by $\delta \in P^+$ have high overlaps with $B^*$ or good fit with its point clouds and vice versa for $P^-$. The loss function to train $f_{boxpc}$ is given by

$$L^{3D}_{boxpc} = \sum_{c \in C_{3D}} L^{c}_{cls} + L^{c}_{reg},$$

(6)

where $L^{c}_{cls}$ is the classification loss

$$L^{c}_{cls} = \begin{cases} H(1, f_{boxpc-cls}(X, B^* - \delta)), & \text{if } \delta \in P^+, \\ H(0, f_{boxpc-cls}(X, B^* - \delta)), & \text{if } \delta \in P^- \end{cases},$$

(7)

to predict the fit of a perturbed 3D box and its frustum point cloud, and $L^{c}_{reg}$ is the regression loss

$$L^{c}_{reg} = l(\delta, f_{boxpc-reg}(X, B^* - \delta)),$$

(8)

to predict the perturbation to correct the perturbed 3D box. $H$ is the binary cross-entropy and $l(y^*, \hat{y})$ is the Smooth L1 loss with $y^*$ as the target for $\hat{y}$.

#### 4.2.2 BoxPC Feature Representations

A possible way to encode the feature representations for the BoxPC network is illustrated on the left of Fig. 4. The features of the input 3D box and frustum point cloud are learned using several MLPs and PointNets [21], respectively. These features are then concatenated and fed into several layers of MLPs to get the BoxPC fit probability $\hat{p}$ and Box Correction term $\Delta \hat{B}$. However, this neglects the intricate information of how the 3D box surfaces cut the

![Figure 4. Different feature representations for a pair of 3D bounding box and point cloud. On the left, the inputs are processed independently and concatenated in the feature space. On the right, the inputs are combined in the input space and processed as a whole.](image-url)
point cloud, leading to poor performance as shown in our ablation studies in Sec. 6.3. Hence, we propose another method to combine the 3D box and point cloud in the input space in a differentiable manner that also exploits the relationship between them. Specifically, we use the box center \((b_x, b_y, b_z)\) to translate the point cloud to the origin to enhance translational invariance to the center positions. Next, we use the box sizes \((b_h, b_w, b_l)\) and rotation \(b_\theta\) to form six planes representing each of the six box surfaces. We compute a \(N \times 6\) feature vector of the perpendicular distance (with direction) from each point to each of the six planes. This feature vector is concatenated with the original frustum point cloud \(X \in \mathbb{R}^{N \times (3+k)}\) to give the box-combined frustum point cloud \(X' \in \mathbb{R}^{N \times (9+k)}\) as illustrated on the right of Fig. 4. This representation allows the network to easily reason about regions where the box surfaces cut the point cloud since the perpendicular distance will be close to 0. Additionally, it is possible to perform feature learning on the 3D box and the point cloud jointly with a single PointNet-based [21] network, improving performance.

4.3. Weak Losses

We supervise the initial 3D box predictions \(\hat{B}_0\) with 2D box labels \(B_{2D}\) and additional priors.

4.3.1 Relaxed Reprojection Loss

Despite the correlation, the 2D box that encloses all the points of a projected 3D box label does not coincide with the 2D box label. We show in Sec. 6.3 that performance deteriorates if we simply minimize the reprojection loss between the 2D box label and the 2D box enclosing the projection of the predicted 3D box onto the image plane. Instead, we propose a relaxed reprojection loss, where boundaries close to the 2D box labels are not penalized. Let \(B_{2D} = [b_{\text{left}}, b_{\text{top}}, b_{\text{right}}, b_{\text{bot}}]\) (green box in Fig. 5) that consists of the left, top, right and bottom image coordinates be the 2D box label with the top-left hand corner as origin. We define an upper bound box \(B_{\text{upper}} = B_{2D} + U\) and a lower bound box \(B_{\text{lower}} = B_{2D} + L\) (outer and inner blue boxes in Fig. 5), where \(U = [u_1, u_2, u_3, u_4]\) and \(L = [v_1, v_2, v_3, v_4]\) are vectors used to adjust the size of \(B_{2D}\). Given an initial 3D box prediction \(\hat{B}_0\), we project it onto the image plane \(\pi(\hat{B}_0)\) (red box in Fig. 5) and obtain an enclosing 2D box around the projected points \(\hat{B}_0^{\text{reproj}} = g(\pi(\hat{B}_0))\) (yellow box in Fig. 5), where \(g(.)\) is the function that returns the enclosing 2D box. We penalize \(\hat{B}_0^{\text{reproj}}\) for violating the bounds specified by \(B_{\text{upper}}\) and \(B_{\text{lower}}\). An example is shown in Fig. 5, the bounds on the left and right are not violated because the left and right sides of the 2D box \(\hat{B}_0^{\text{reproj}}\) stays within the bounds specified by \(B_{\text{upper}}\) and \(B_{\text{lower}}\) (blue boxes). However, the top and bottom of \(\hat{B}_0^{\text{reproj}}\) beyond the bounds are penalized. More formally, the relaxed reprojection loss is given by

\[
L_{2D}^{\text{reproj}} = \sum_{c \in \mathcal{C}_{2D}} \sum_{i=1}^{4} l(v_i, u_i, \hat{B}_0^{c,\text{reproj}}(i)),
\]

where \(l(y_{\text{lower}}, y_{\text{upper}}, \hat{y}) = \begin{cases} l(y_{\text{upper}}, \hat{y}), & \text{if } \hat{y} > y_{\text{upper}}, \\ l(y_{\text{lower}}, \hat{y}), & \text{if } \hat{y} < y_{\text{lower}}, \\ 0 & \text{otherwise.} \end{cases} \)

(10)

\(\hat{B}_0^{c,\text{reproj}}(i)\) retrieves the \(i\)-th component of \(\hat{B}_0^{c,\text{reproj}}\).

4.3.2 Box Prior Loss

We use prior knowledge on the object volume and size to regularize the training loss. More specifically, we add the prior loss

\[
L_{2D}^{\text{prior}} = \sum_{c \in \mathcal{C}_{2D}} L_{\text{vol}}^c + L_{s-\text{var}}^c,
\]

(11)

to train our network, where

\[
L_{\text{vol}}^c = \max(0, V^c - \hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c),
\]

(12)

is to penalize predictions with volumes that are below class-specific thresholds from prior knowledge about the scale of objects, and

\[
L_{s-\text{var}}^c = l(\hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c) + l(\hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c),
\]

(13)

is to penalize the size variance of the predictions within each class in each minibatch since there should not be excessive size variations within a class. \((\hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c)\) are the size predictions from \(\hat{B}_0\) for class \(c\), and \((\hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c, \hat{b}_0^c)\) are the average size predictions per minibatch for class \(c\) and \(V^c\) is the volume threshold for class \(c\).

5. Training

We train a Faster RCNN-based [22] 2D object detector with the 2D box labels from classes \(C\) and train the BoxPC Fit Network by minimizing \(L_{\text{boxpc}}\) from Eq. 6 with the 3D
box labels from classes \(C_{3D}\). Finally, when we train the 3D object detector, we alternate between optimizing the losses for classes in \(C_{2D}\) and classes in \(C_{3D}\). Specifically, when optimizing for \(C_{2D}\), we train \(f_{\text{box}}\) to minimize

\[
L_{\text{box}}^{2D} = L_{\text{fit}}^{2D} + L_{\text{reproj}}^{2D} + L_{\text{prior}}^{2D},
\]

where \(L_{\text{fit}}^{2D}\), \(L_{\text{reproj}}^{2D}\) and \(L_{\text{prior}}^{2D}\) are from Eq. 5, 9 and 11 respectively. When optimizing for \(C_{3D}\), we train \(f_{\text{seg}}\) and \(f_{\text{box}}\) to minimize \(L_{\text{seg}}^{3D}\) and \(L_{\text{box}}^{3D}\) from Eq. 1 and 2 respectively. Hence, the loss functions for \(f_{\text{seg}}\) and \(f_{\text{box}}\) are respectively given by

\[
L_{\text{seg}} = L_{\text{seg}}^{3D},
\]

\[
L_{\text{box}} = L_{\text{box}}^{3D} + L_{\text{box}}^{2D}.
\]

6. Experiments

6.1. Datasets

We evaluate our model on SUN-RGBD and KITTI benchmarks for 3D object detection. For the SUN-RGBD benchmark, we evaluate on 10 classes as in [20]. To test the performance of cross-category semi-supervised 3D object detection (CS3D), we randomly split the 10 classes into 2 subsets \((C_A\) and \(C_B)\) of 5 classes each (the 5 classes on the left and on the right of Tab. 1). First, we let \(C_A\) be the strong classes and \(C_B\) be the weak classes by setting \(C_{3D} = C_A\), \(C_{2D} = C_B\) to obtain the evaluation results for \(C_B\). Next, we reversed \(C_A\) and \(C_B\) to obtain the evaluation results for \(C_A\). We follow the same \(train/val\) split and performance measure of average precision (AP) with 3D Intersection-over-Union (IoU) of 0.25 as in [20].

The KITTI benchmark evaluates on Cars, Pedestrians and Cyclists classes. To test the CS3D setting, we set any 2 classes to \(C_{3D}\) and the remaining class to \(C_{2D}\) to obtain the evaluation results on \(C_{2D}\). For example, to obtain the evaluation results on Cyclists, we set \(C_{3D} = \{\text{Cars, Pedestrians}\}\). We follow the same \(train/val\) split as in [20] and measure performance using AP with 3D IoU of 0.25 for all classes. For fully-supervised setting, we use 3D IoU of 0.7 for Cars and 0.5 for Pedestrians and Cyclists.

6.2. Comparison with Baselines

To the best of our knowledge, there are no other methods that demonstrate weakly- or semi-supervised 3D object detection, therefore we design baselines (details in supplementary) with the state-of-the-art FPN [20] where it trains on strong labels from \(C_{3D}\) and tests on \(C_{2D}\). Specifically, we use the original network (“FPN*”) and a network without the one hot class vector (“FPN* w/o OneHot”). The latter performs better since there are no 3D box labels for classes \(C_{2D}\). Finally, we train fully-supervised BoxPC Fit networks to further improve the fully-supervised FPN [20].

SUN-RGBD In Tab. 1, BoxPC, R, P refer to the usage of the proposed BoxPC Fit network, relaxed reprojection loss and box prior loss respectively.

<table>
<thead>
<tr>
<th>Fully-Supervised:</th>
<th>bathtub</th>
<th>bed</th>
<th>toilet</th>
<th>chair</th>
<th>desk</th>
<th>dresser</th>
<th>nstand</th>
<th>sofa</th>
<th>table</th>
<th>bkshelf</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSS [26]</td>
<td>44.2</td>
<td>78.8</td>
<td>78.9</td>
<td>61.2</td>
<td>20.5</td>
<td>6.4</td>
<td>15.4</td>
<td>53.5</td>
<td>50.3</td>
<td>11.9</td>
<td>42.1</td>
</tr>
<tr>
<td>COG [23]</td>
<td>58.3</td>
<td>63.7</td>
<td>70.1</td>
<td>62.2</td>
<td>45.2</td>
<td>15.5</td>
<td>27.4</td>
<td>51.0</td>
<td>51.3</td>
<td>31.8</td>
<td>47.6</td>
</tr>
<tr>
<td>2D-driven [13]</td>
<td>43.5</td>
<td>64.5</td>
<td>80.4</td>
<td>48.3</td>
<td>27.9</td>
<td>25.9</td>
<td>41.9</td>
<td>50.4</td>
<td>37.0</td>
<td>31.4</td>
<td>45.1</td>
</tr>
<tr>
<td>LSS [24]</td>
<td>76.2</td>
<td>73.2</td>
<td>73.7</td>
<td>60.5</td>
<td>34.5</td>
<td>13.5</td>
<td>30.4</td>
<td>60.4</td>
<td>55.4</td>
<td>32.9</td>
<td>51.0</td>
</tr>
<tr>
<td>FPN [20]</td>
<td>43.3</td>
<td>81.1</td>
<td>90.9</td>
<td>64.2</td>
<td>24.7</td>
<td>32.0</td>
<td>58.1</td>
<td>61.1</td>
<td>51.1</td>
<td>33.3</td>
<td>54.0</td>
</tr>
<tr>
<td>FPN*</td>
<td>46.0</td>
<td>80.3</td>
<td>82.8</td>
<td>58.5</td>
<td>24.2</td>
<td>36.9</td>
<td>47.6</td>
<td>54.9</td>
<td>42.0</td>
<td>41.6</td>
<td>51.5</td>
</tr>
<tr>
<td>FPN* + BoxPC Refine</td>
<td>51.5</td>
<td>81.0</td>
<td>85.0</td>
<td>59.0</td>
<td>24.5</td>
<td>38.6</td>
<td>52.2</td>
<td>55.3</td>
<td>43.8</td>
<td>40.9</td>
<td>53.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CC Semi-Supervised:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FPN*</td>
<td>6.1</td>
<td>69.5</td>
<td>28.1</td>
<td>12.4</td>
<td>18.1</td>
<td>17.5</td>
<td>37.4</td>
<td>25.8</td>
<td>23.5</td>
<td>6.9</td>
<td>24.5</td>
</tr>
<tr>
<td>FPN* w/o OneHot</td>
<td>24.4</td>
<td>69.5</td>
<td>30.5</td>
<td>15.9</td>
<td>22.6</td>
<td>19.4</td>
<td>39.0</td>
<td>37.3</td>
<td>29.0</td>
<td>12.8</td>
<td>30.1</td>
</tr>
<tr>
<td>Ours + R</td>
<td>29.5</td>
<td>60.9</td>
<td>65.3</td>
<td>36.0</td>
<td>20.2</td>
<td>27.3</td>
<td>50.9</td>
<td>46.4</td>
<td>28.4</td>
<td>6.7</td>
<td>37.2</td>
</tr>
<tr>
<td>Ours + BoxPC</td>
<td>28.4</td>
<td>67.9</td>
<td>73.3</td>
<td>32.3</td>
<td>23.3</td>
<td>31.0</td>
<td>50.9</td>
<td>48.9</td>
<td>33.7</td>
<td>16.4</td>
<td>40.6</td>
</tr>
<tr>
<td>Ours + BoxPC + R</td>
<td>28.4</td>
<td>68.1</td>
<td>77.9</td>
<td>32.9</td>
<td>23.3</td>
<td>30.6</td>
<td>51.1</td>
<td>49.9</td>
<td>34.7</td>
<td>13.7</td>
<td>41.1</td>
</tr>
<tr>
<td>Ours + BoxPC + R + P</td>
<td>30.2</td>
<td>70.7</td>
<td>76.3</td>
<td>33.6</td>
<td>24.0</td>
<td>32.5</td>
<td>52.0</td>
<td>49.8</td>
<td>34.2</td>
<td>14.8</td>
<td>41.8</td>
</tr>
</tbody>
</table>

Table 1. 3D object detection AP on SUN-RGBD val set. Fully-supervised methods are trained on 2D and 3D box labels of all classes while Cross-category (CC) Semi-supervised methods are trained on 3D box labels of classes in \(C_{3D}\) and 2D box labels of all classes. BoxPC, R, P refer to the BoxPC Fit network, relaxed reprojection loss and box prior loss respectively. * refers to our implementation.
58.4% for baseline “FPN* w/o OneHot”). We consider this a promising result for semi-supervised methods, and it demonstrates the effectiveness of our proposed model in transferring knowledge to unseen classes. In addition, we show the effectiveness of the BoxPC Fit network in fully-supervised settings by training the BoxPC Fit network on all classes \( C \) and using the network to refine the 3D box predictions of the state-of-the-art FPN [20]. As seen in Tab. 1, “FPN* + BoxPC Refine” outperforms the vanilla fully-supervised “FPN*” in every single class except for bookshelf, showing the usefulness of the BoxPC Fit network even in fully-supervised settings.

**KITTI** In Tab. 2, the baselines are unable to achieve any performance on **Cars** when trained on **Pedestrians** and **Cyclists** because of huge differences in sizes. The network simply assumes a **Car** instance is small and makes small predictions. With the addition of a prior volume, it becomes possible to make predictions on **Cars**. We observe that adding “BoxPC + R + P” significantly improves performance over the baseline’s mean AP of 0.07% to 59.95% for **Cars**. It also improved the baseline’s performance from 48.03% to 67.49% for **Pedestrians** and from 41.89% to 51.83% for **Cyclists**. Similarly, we show that a fully-supervised BoxPC Fit network is able to refine and improve the 3D box predictions made by “FPN” in Tab. 3. In this case, “FPN + BoxPC Refine” improves AP for all classes at all difficulty levels over the original “FPN”.

### 6.3. Ablation Studies

**Relaxed Reprojection Loss** To illustrate the usefulness of using a relaxed version of the reprojection loss, we vary the lower and upper bounds \( B_{lower}^* \) and \( B_{upper}^* \). We set \( B_{lower}^*, B_{upper}^* \) to different scales of \( B_{2D}^* \), i.e. 1.5 \( B_{2D}^* \) refers to the 2D box centered at the same point as \( B_{2D}^* \) but with 1.5× size. By setting \( B_{lower}^*, B_{upper}^* = B_{2D}^* \), it reverts from the relaxed version to the direct version of reprojection loss. We train with similar settings as “Ours + R” in Tab. 1, and show in Tab. 4 that mean AP improves from 25.6% \( (B_{upper}^* = B_{2D}^* ) \) to 37.2% \( (B_{upper}^* = 1.5 B_{2D}^* ) \).

**BoxPC Fit Network Objectives** The BoxPC Fit network is trained on classification and regression objectives derived from the 3D box labels. To show the effectiveness of having both objectives, we adopt different settings in which classification or regression objectives are active. We train with similar settings as “Ours + BoxPC” in Tab. 1. Without the classification objective for the BoxPC Fit network, the 3D object detector is unable to maximize \( \hat{p} = f_{boxpc—cls}(X, B_0) \) during training. Omission of the regression objective prevents the initial box predictions \( B_0 \) from being refined further by \( \Delta B = f_{boxpc—reg}(X, B_0) \) to the final box predictions \( B \). From Tab. 5, we see that training without either classification or regression objective causes performance to drop from 40.6% to 32.1% or 39.8%.

**BoxPC Representation** To demonstrate the importance of a joint Box to Point Cloud input representation as seen on the right of Fig. 4, we train on the “Ours + BoxPC” setting in Tab. 1 with different BoxPC representations. As shown in Tab. 5, the performance of combined learning of BoxPC features exceeds the performance of independent learning of BoxPC features by 6.2%.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cars Easy</th>
<th>Moderate</th>
<th>Hard</th>
<th>Pedestrians Easy</th>
<th>Moderate</th>
<th>Hard</th>
<th>Cyclists Easy</th>
<th>Moderate</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fully-Supervised:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPN [20]</td>
<td>96.69</td>
<td>96.29</td>
<td>88.77</td>
<td>85.40</td>
<td>79.20</td>
<td>71.61</td>
<td>86.59</td>
<td>67.60</td>
<td>63.95</td>
</tr>
<tr>
<td>CC Semi-Supervised:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPN</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>4.31</td>
<td>4.37</td>
<td>3.74</td>
<td>1.05</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td>FPN w/o OneHot</td>
<td>0.08</td>
<td>0.07</td>
<td>0.07</td>
<td>55.66</td>
<td>47.33</td>
<td>41.10</td>
<td>54.35</td>
<td>36.46</td>
<td>34.85</td>
</tr>
<tr>
<td>Ours + BoxPC + R</td>
<td>14.71</td>
<td>12.19</td>
<td>11.21</td>
<td>75.13</td>
<td>62.46</td>
<td>57.18</td>
<td>61.25</td>
<td>41.96</td>
<td>40.07</td>
</tr>
<tr>
<td>Ours + BoxPC + R + P</td>
<td>69.78</td>
<td>58.66</td>
<td>51.40</td>
<td>76.85</td>
<td>66.92</td>
<td>58.71</td>
<td>63.29</td>
<td>47.54</td>
<td>44.66</td>
</tr>
</tbody>
</table>

Table 2. 3D object detection AP on KITTI val set, evaluated at 3D IoU threshold of 0.25 for all classes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cars Easy</th>
<th>Moderate</th>
<th>Hard</th>
<th>Pedestrians Easy</th>
<th>Moderate</th>
<th>Hard</th>
<th>Cyclists Easy</th>
<th>Moderate</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fully-Supervised:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VeloFCN [14]</td>
<td>15.20</td>
<td>13.66</td>
<td>15.98</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MV3D [6]</td>
<td>71.29</td>
<td>62.68</td>
<td>56.56</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Voxelnet [40]</td>
<td>89.60</td>
<td>84.81</td>
<td>78.57</td>
<td>65.95</td>
<td>61.05</td>
<td>56.98</td>
<td>74.41</td>
<td>52.18</td>
<td>50.49</td>
</tr>
<tr>
<td>FPN [20]</td>
<td>84.40</td>
<td>71.37</td>
<td>63.38</td>
<td>65.80</td>
<td>56.40</td>
<td>49.74</td>
<td>76.11</td>
<td>56.88</td>
<td>53.17</td>
</tr>
<tr>
<td>FPN + BoxPC Refine</td>
<td>85.63</td>
<td>72.10</td>
<td>64.25</td>
<td>68.83</td>
<td>59.41</td>
<td>52.08</td>
<td>78.77</td>
<td>58.41</td>
<td>54.52</td>
</tr>
</tbody>
</table>

Table 3. 3D object detection AP on KITTI val set, evaluated at 3D IoU threshold of 0.7 for **Cars** and 0.5 for **Pedestrians** and **Cyclists**.
Table 4. 3D object detection AP on SUN-RGBD val set with varying $B_{upper}^*$ for the relaxed reprojection loss function.

<table>
<thead>
<tr>
<th>$B_{lower}^*$</th>
<th>$B_{upper}^*$</th>
<th>bathtub</th>
<th>bed</th>
<th>toilet</th>
<th>chair</th>
<th>desk</th>
<th>dresser</th>
<th>nstand</th>
<th>sofa</th>
<th>table</th>
<th>bkshelf</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_{2D}^*$</td>
<td>1.0 $B_{2D}^*$</td>
<td>17.0</td>
<td>11.9</td>
<td>60.1</td>
<td>24.3</td>
<td>11.2</td>
<td>24.0</td>
<td>46.8</td>
<td>30.3</td>
<td>19.2</td>
<td><strong>11.3</strong></td>
<td>25.6</td>
</tr>
<tr>
<td>$B_{2D}^*$</td>
<td>1.5 $B_{2D}^*$</td>
<td><strong>29.5</strong></td>
<td>60.9</td>
<td><strong>65.3</strong></td>
<td><strong>36.0</strong></td>
<td>20.2</td>
<td><strong>27.3</strong></td>
<td><strong>50.9</strong></td>
<td>46.4</td>
<td>28.4</td>
<td>6.7</td>
<td><strong>37.2</strong></td>
</tr>
<tr>
<td>$B_{2D}^*$</td>
<td>2.0 $B_{2D}^*$</td>
<td>16.9</td>
<td>23.4</td>
<td>55.8</td>
<td>35.0</td>
<td>19.6</td>
<td>25.1</td>
<td>49.2</td>
<td>48.3</td>
<td><strong>31.3</strong></td>
<td>3.5</td>
<td>30.8</td>
</tr>
</tbody>
</table>

Table 5. 3D object detection AP on SUN-RGBD val set when BoxPC Fit Network is trained on different representations and objectives.

<table>
<thead>
<tr>
<th>Features</th>
<th>Cls</th>
<th>Reg</th>
<th>bathtub</th>
<th>bed</th>
<th>toilet</th>
<th>chair</th>
<th>desk</th>
<th>dresser</th>
<th>nstand</th>
<th>sofa</th>
<th>table</th>
<th>bkshelf</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combined</td>
<td>✓</td>
<td>✓</td>
<td>25.9</td>
<td>67.3</td>
<td>71.8</td>
<td>30.6</td>
<td><strong>23.6</strong></td>
<td>31.8</td>
<td>48.8</td>
<td>48.1</td>
<td>32.6</td>
<td><strong>17.3</strong></td>
<td>39.8</td>
</tr>
<tr>
<td>Combined</td>
<td>✓</td>
<td>✓</td>
<td><strong>27.4</strong></td>
<td><strong>69.4</strong></td>
<td>39.8</td>
<td>17.7</td>
<td>23.1</td>
<td>18.5</td>
<td>39.9</td>
<td>41.7</td>
<td>30.2</td>
<td>12.9</td>
<td>32.1</td>
</tr>
<tr>
<td>Independent</td>
<td>✓</td>
<td>✓</td>
<td><strong>39.6</strong></td>
<td>68.1</td>
<td>24.5</td>
<td>26.8</td>
<td>22.3</td>
<td><strong>34.4</strong></td>
<td>38.8</td>
<td>46.7</td>
<td>30.3</td>
<td>12.3</td>
<td>34.4</td>
</tr>
<tr>
<td>Combined</td>
<td>✓</td>
<td>✓</td>
<td>28.4</td>
<td>67.9</td>
<td><strong>73.3</strong></td>
<td><strong>32.3</strong></td>
<td>23.3</td>
<td>31.0</td>
<td><strong>50.9</strong></td>
<td><strong>48.9</strong></td>
<td><strong>33.7</strong></td>
<td>16.4</td>
<td><strong>40.6</strong></td>
</tr>
</tbody>
</table>

Figure 6. Qualitative comparisons between the baseline, fully-supervised and proposed models on the SUN-RGBD val set.

6.4. Qualitative Results

We visualize some of the predictions by the baseline, fully-supervised and proposed models in Fig. 6. We used 2D box labels for frustum proposals to reduce noise due to 2D detections. We observe that the proposed model’s predictions are closer to the fully-supervised model than the baseline’s. The baseline’s predictions tend to be inaccurate since it is not trained to fit the objects from weak classes.

7. Conclusion

We propose a transferable semi-supervised model that is able to perform 3D object detection on weak classes with only 2D box labels. Our method achieves strong performance over the baselines and improves the practicality and usefulness of 3D object detectors in new applications.

Acknowledgement This work is supported in part by a Singapore MOE Tier 1 grant R-252-000-A65-114.
References


