LABEL-FREE SYNTHETIC PRETRAINING OF OBJECT DETECTORS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a new approach, Synthetic Optimized Layout with Instance Detection (SOLID), to pretrain object detectors with synthetic images. Our "SOLID" approach consists of two main components: (1) generating synthetic images using a collection of unlabelled 3D models with optimized scene arrangement; (2) pretraining an object detector on "instance detection" task—given a query image depicting an object, detecting all instances of the exact same object in a target image. Our approach does not need any semantic labels for pretraining and allows the use of arbitrary, diverse 3D models. Experiments on COCO show that with optimized data generation and a proper pretraining task, synthetic data can be highly effective data for pretraining object detectors. In particular, pretraining on rendered images achieves performance competitive with pretraining on real images while using significantly less computing resources.

1 INTRODUCTION

Object detection is a key computer vision task. Currently, state-of-the-art systems are trained on a large number of manually annotated images. However, manual annotations are costly to acquire; as a result, such reliance can potentially become a bottleneck for further improvement. An important question is whether additional performance gains can be achieved by using alternative sources of data without manual labels.

To reduce the dependency on manual labels, recent work (He et al., 2020; Chen et al., 2020d;b; Hénaff et al., 2021; Roh et al., 2021; Xiao et al., 2021; Bar et al., 2021; Caron et al., 2020; Ye et al., 2019; He et al., 2021; Pathak et al., 2016; Vincent et al., 2008; 2010; Bao et al., 2021; Chen et al., 2020a; Misra & van der Maaten, 2020; Kolesnikov et al., 2019; Doersch & Zisserman, 2017; Cao et al., 2020; Chen & He, 2021; Asano et al., 2020; Caron et al., 2018; Huang et al., 2019) has explored self-supervised pretraining, which leverages the massive amounts of unlabeled images online. Common approaches of self-supervision include contrastive learning (He et al., 2020; Chen et al., 2020d;b; Hénaff et al., 2021; Roh et al., 2021; Xiao et al., 2021; Bar et al., 2021; Caron et al., 2020; Ye et al., 2019), where a network learns features invariant to known 2D image augmentations, and reconstructive learning (He et al., 2021; Pathak et al., 2016; Vincent et al., 2008; 2010; Bao et al., 2021; Chen et al., 2020a), where a network learns to predict missing/masked parts of data using the rest.

Despite many promising results, self-supervised pretraining still faces significant technical challenges. Contrastive approaches heavily depend on effective image augmentations, which can be difficult to design beyond a few simple 2D transforms. Reconstructive approaches can be highly sensitive to the relevance of the reconstruction task to downstream applications. For example, reconstructing raw pixel intensities may not be as useful for object detection where invariance to intensity changes is important.

On the other hand, synthetic data has been widely used to supplement real images for many computer vision tasks, with notable successes in 3D vision (Mayer et al., 2016; Butler et al., 2012; Tremblay et al., 2018b; Wang et al., 2020a; Lipson et al., 2021; Zhang et al., 2020; 2019; Teed & Deng, 2021; Labbé et al., 2020). However, using synthetic data has yet to become common practice for object detection, except in specialized domains such as autonomous driving (Tremblay et al., 2018a; Johnson-Roberson et al., 2017; Alhaija et al., 2017; Prakash et al., 2021). One possible reason for this discrepancy is that for synthetic data to work well, they should closely approximate the real data (i.e. small real-sim domain gap), but for object detection it is difficult to generate synthetic data that matches real data in terms of the diversity of objects and scenes. For example, it would be

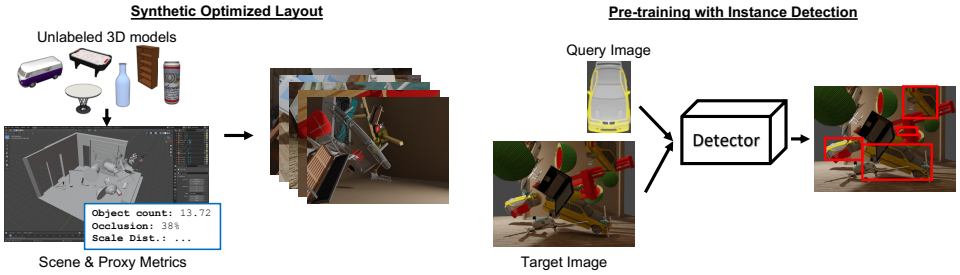


Figure 1: Overview of our approach. We generate synthetic images using unlabeled 3D models with optimized scene layout. We then pretrain an object detector to perform instance detection: given a query object, detect all instances in a target image.

extremely challenging to create a synthetic dataset to cover all 80 object categories in the COCO benchmark (Lin et al., 2014), including realistic compositions of scenes and varied instances of each object category (e.g. varied dogs and cats). Thus it is perhaps expected that outside specialized domain, synthetic data would have a limited role to play for object detection.

In this work, we present a new result that defies conventional wisdom: synthetic data can be surprisingly effective for object detection, even with limited diversity and realism. In particular, we can generate effective synthetic data using only a collection of 3D models *without any category labels*; pretraining on such synthetic data achieves results competitive to self-supervised pretraining on real images.

We achieve this result through "Synthetic Optimized Layout with Instance Detection (SOLID)", a new approach we introduce for pretraining object detectors. Our approach, as illustrated in Fig. 1, consists of two main ingredients: (1) generating synthetic images using a collection of unlabeled 3D models, with optimized scene arrangement; (2) pretraining an object detector on the "instance detection" task (Mercier et al., 2021; Ammirato et al., 2018)—given a query image depicting an object, detecting all instances of the exact same object in a target image.

It is worth noting that our approach only uses *unlabeled* 3D models. That is, no semantic labels are necessary; in fact, the 3D models can be arbitrary shapes that do not fit into any known object category. Doing away with semantic labels gets around the difficulty in generating 3D shapes that conform to semantic categories (e.g. generating varied and realistic 3D shapes of cats is an unsolved problem), allowing the use of arbitrary 3D shapes which can significantly expand the diversity of synthetic data. Coupled with this label-free synthetic data is the instance detection pretraining task, which, unsurprisingly, also requires no semantic labels. The task is posed purely geometrically—finding in a target image all instances of the 3D shape depicted in a query image.

Our instance detection task differs from the standard object detection in that the input consists of two images, a query image that specifies the object to detect, and a target image in which to detect the instances of the object. The object specified by the query image may be completely novel and may have never been seen during training. In other words, unlike standard object detection, there is no fixed, predetermined list of objects or object classes to detect. During test time, the network needs to detect all the instances given just a single example of a new object.

Our "SOLID" approach is motivated by the hypothesis that a significant source of difficulty of object detection is geometrical invariances, particularly invariances to occlusion, illumination, and viewpoint. Such invariances are likely learnable independent of category labels—an infant may not know that the object she likes to play with is called a "toy car", but she can almost certainly find it in a photo effortlessly. This hypothesis naturally leads to the instance detection task, which focuses the pretraining on learning geometric invariances.

Given a collection of unlabeled 3D models, there are infinitely many possible synthetic images one can generate, but only a finite subset of them can be used during pretraining and not all subsets are equally useful. We thus optimize the parameters and design choices of the rendering pipeline to maximize the effectiveness of the synthetic data. This primarily involves the spatial layout of the objects in front of the camera. A naive approach would be to try each possible layout, render a dataset, pretrain a model, fine-tune on labeled data, and evaluate performance on a validation set of real images. But this would be too expensive to be feasible. Instead, we optimize the layout against

a set of proxy metrics of scene complexity including object count, amount of occlusion, and scale distribution, which we find to be good indicators of the validation performance on real images.

The main novelty of work is a new pretraining method of object detection that integrates two existing ideas: synthetic data and instance detection. Neither synthetic data nor instance detection is novel on its own, but to our knowledge no prior work has combined the two for pretraining object detection. Our approach has unique advantages over existing alternatives. Compared to existing methods using synthetic data, our approach does not require semantic labels, which means that arbitrary 3D shapes can be used. Compared to existing methods using real images, our approach allows 3D data augmentations that are difficult to achieve with real images.

We evaluate our approach on the standard COCO (Lin et al., 2014) dataset. We generate synthetic images using 3D models from ShapeNet (Chang et al., 2015) and SceneNet (Handa et al., 2015), and pretrain standard object detectors including Faster-RCNN (Ren et al., 2017) and Mask-RCNN (He et al., 2019). Experiments on COCO show that pretraining on rendered images achieves performance competitive with pretraining on real images, including MoCov3 (Chen et al., 2021), DetCon (Hénaff et al., 2021) and SCRL (Roh et al., 2021), while using significantly less computing resources. Our results demonstrate that with our novel combination of optimized data generation and a proper pretraining task, synthetic data can be highly effective data for pretraining object detectors.

2 Related Work

Supervised and Unsupervised pretraining for Object Detection Pretraining visual models including object detectors has a long history. Early work (Hinton et al., 2006; Ranzato et al., 2006; Bengio et al., 2007) shows that pretraining each layer with an unsupervised learning algorithm before fine-tuning the network improves the network's performance significantly. A similar approach is also used in a pedestrian detector (Sermanet et al., 2013). Later, R-CNN (Girshick et al., 2014) shows that supervised pretraining on the ImageNet classification dataset (Deng et al., 2009) significantly improves detection performance. Since then, pretraining on the image classification task has become a common practice in state-of-the-art object detectors (Zhou et al., 2021; Sun et al., 2021; Tan et al., 2020).

Recent approaches (He et al., 2020; Chen et al., 2020d; 2021; 2020b;c; Caron et al., 2020) based on contrastive learning (Hadsell et al., 2006) has achieved competitive performance against supervised pretraining on various downstream tasks. In contrastive learning, a network learns to predict similar embeddings for augmented views of the same image. But these approaches rely on image-level features to predict the embeddings, which is not ideal for object detection. Newer approaches have proposed new pretraining tasks that focus on the object-level features. SCRL (Roh et al., 2021), ReSim-FPN$^{\mathrm{T}}$ (Xiao et al., 2021) and DetCo (Xie et al., 2021) train a network to predict similar embedding vectors between patches of the two augmented views. Instead of patches, DetCon (Hénaff et al., 2021) uses regions that are segmented by a graph-based algorithm. InsLoc (Yang et al., 2021) randomly crops two overlapping patches from an image, paste them onto different images and train a network to predict similar embedding vectors between the patches.

UP-DETR (Dai et al., 2021) proposes a pretraining task where random patches are extracted from an image and a network is trained to localize them in the same image. DETReg (Bar et al., 2021) proposes to train an object detector to predict bounding boxes generated from the selective search algorithm (Uijlings et al., 2013) and to mimic the output of a network trained with contrastive learning.

Our approach is similar to these existing approaches in that our pretraining task can be thought of as a form of contrastive learning—predicting embeddings to distinguish image regions of the same object from other regions. But our approach differs in that we use synthetic images instead of real images. Using synthetic images allows us to easily generate augmented versions of the same object with occlusion and viewpoint change. Such augmentations would be very difficult to generate for natural images.

Instance Detection The instance detection task has also been studied by prior work. Ammirato et al. (Ammirato et al., 2018) and Mercier et al. (Mercier et al., 2021) suggest this task is useful for robotics and augmented reality applications which often require recognizing a very specific instance, and propose different approaches to the task. Compared to these works, the novelty of our work is in integrating synthetic data and the instance detection task to pretrain object detectors.

Figure 2: The first row shows query images and the second row shows target images.

Synthetic Training of Object Detectors Prior work has also studied how to use synthetic data to train an object detector. Peng et al. (Peng et al., 2015), Alhaija et al. (Alhaija et al., 2017), Hinterstoisser et al. (Hinterstoisser et al., 2018) and Tremblay et al. (Tremblay et al., 2018a) generate the synthetic data by rendering 3D models over a real world background scene. The synthetic data are then used to fine-tune networks pretrained on ImageNet.

Other work has obtained training data from computer games for object detection and semantic segmentation. Bochinski et al. (Bochinski et al., 2016) use Garry's Mod to train a detector for cars, persons and animals. Richter et al. (Richter et al., 2016) obtain data for semantic segmentation in Grand Theft Auto. Shafaei et al. (Shafaei et al., 2016) show that a network pretrained on synthetic data collected from a game outperforms network pretrained on real world data after fine-tuning. Data extracted from the games may not come with foreground or background labels. So extra steps such as background subtraction (Bochinski et al., 2016) or manual labelling (Richter et al., 2016) are needed to identify foreground objects or pixels.

Our method also uses synthetic data, but unlike these existing approaches, we introduce a new pretraining task. All of these prior works perform pretraining in the form of standard object detection—the input consists of a single image and the task is detect objects in a fixed, pre-defined; as a result, the network needs to memorize through training what each object in the list looks like. In contrast, our instance detection pretraining task has two input images (a query image and a target image), and the network only needs to learn geometric invariances as opposed to the object-specific visual features which are less transferrable to new domains.

3 Approach

Our "SOLID" approach involves two main ingredients: generating synthetic images and pretraining a network on instance detection–given a query image of an unknown object, detecting all instances of the same object in a target image. After pretraining, the pretrained object detector is fine-tuned on a downstream dataset. Fig. 1 gives an overview of SOLID.

3.1 Generating Synthetic Images

We generate our synthetic images by placing 3D models from ShapeNet (Chang et al., 2015) into backgrounds from SceneNet (Handa et al., 2015). There is a large space of possible layouts of the objects relative to the camera, but some of them are likely to be more useful than others as synthetic data. We can find out the usefulness of a subset of layouts by using them to pretrain a detector and evaluate the validation performance on real images, but this would be prohibitively expensive. Instead, we propose a set of proxy metrics of scene complexity to guide data generation. These proxy metrics can be evaluated quickly without going through actual pretraining and capture the known failure cases of object detection (crowded scenes, occlusion, small objects, viewpoint change, etc.). Empirically we find these metrics to correlate well with the final validation performance.

Proxy Metrics We use the following proxy metrics of scene complexity:

- **Average occlusion:** For each object, in addition to the usual segmentation mask m_p used in the instance segmentation task, we render a segmentation mask m_f by hiding other objects in the scene. The occlusion of an object is defined as $(n_f - n_p)/n_f$ where n_p is the number of pixels in m_p and n_f is the number of pixels in m_f.

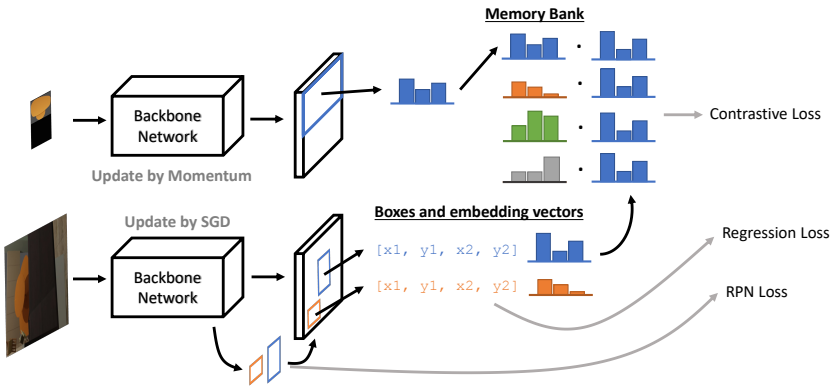


Figure 3: Our network architecture consists of two detectors. The first detector is applied to a query image to extract an embedding vector for an object, which will be stored in a memory bank, and the second detector is applied to a target image to predict boxes and an embedding vector for each box. We calculate the dot products between an embedding vector of a region to all of the embedding vectors in the memory bank and apply contrastive loss, in addition to the standard regression loss and RPN loss. The first detector is updated by momentum while the second one is updated by SGD.

- **Scale distributions:** Following the definition in COCO, we divide the objects into small, medium and large objects by the number of pixels in their segmentation masks. We then calculate the percentage of each scale.
- **Object count:** This is the number of visible objects in the rendered image. Some objects may not be visible in the rendered image because they may be completely occluded.

Rendering Pipeline The scenes from SceneNet come with 3D models. We remove all 3D models in the scenes except the ceiling, floor and wall before they are used as backgrounds.

After initializing the background, we randomly place a camera away from the center of the background and point it toward the central area of the background to allow enough space for placing the 3D models. Because there may be walls in the background, we perform ray casting to find out the nearest obstacle to the camera. If it is too close, we sample another location until there is enough space to place the objects. Only the locations within the viewing frustum is considered in the subsequent steps, which improves efficiency and ensures that at least part of the objects can be visible in the final images. We randomly select, scale, and rotate a 3D model and place it at a location where its 3D bounding box does not intersect with 3D bounding boxes from other models.

In our rendering pipeline, the spatial layout of objects is influenced by several parameters and design choices, which we optimize against our proxy metrics. Instead of randomly placing a 3D model, we place it in front of or behind other 3D models with respect to the camera to increase the amount of occlusion. To increase the object count, we pack more objects into a scene by allowing them to be floating in the air. We optimize the scales and the rotations of the 3D models to have different scale distributions and include more viewpoints for each object.

After placing all the objects, we apply three point lighting to a random object in the scene to illuminate the scene before we render the image. In all of our rendered datasets, we do not run any physics simulation so that we have complete control over the layout. We also render multiple query images for each object to be used in our instance detection task. To render a query image, we place an object into an empty background, rotate it along the z-axis and render an image every 45 degrees. Fig. 2 shows some example target and query images.

3.2 Pretraining with Instance Detection

Given a query image depicting an object, our instance detection task is to detect the exact same instances of the object in the target image.

Given an existing object detector, we propose a "wrapper" architecture to pretrain an object detector on our instance detection task as shown in Fig. 3. Our wrapper architecture consists of two identical

object detectors, one for the query image and one for the target image. The first detector predicts an embedding vector for the query image, while the second detector detects objects in the target image and predicts an embedding vector for each predicted box. The embedding vectors should be similar if they depict the same object. We use Faster R-CNN and Mask R-CNN as our choice of detectors but other detectors such as one-stage detectors can be used in this wrapper architecture. Below, we use Faster R-CNN as an example to describe the details.

The first Faster R-CNN extracts an embedding vector q_j from a query image depicting an object j and does not predict any box. The query image is padded so that it can be fed into our network. We then use an RoIAlign layer to extract features for the object to not include features from the padded area. We also replace the classification head in this Faster R-CNN with a fully-connected layer with 256 channels which predicts an embedding vector from the features. Neither the RPN nor the bounding box regressor in this Faster R-CNN is used. Inspired by MoCo (He et al., 2020), we have a memory bank that stores an embedding vector for every object.

There is a detail worth mentioning for updating the memory bank. We use multiple GPUs to train our detectors and each GPU has its own data sampling process. Each process chooses n query images for its batch of target images independently. If there are more than n unique objects in the target images, it randomly chooses n of them and picks one random query image for each chosen object. Otherwise it samples from objects that are not in the target images. As a result each GPU uses different query images which creates inconsistency between memory banks of different GPUs. So before updating its memory bank, each GPU gathers the embedding vectors from other GPUs.

The second Faster R-CNN detects objects in target image. Similar to a conventional Faster R-CNN, it first predicts a set of region of interests (RoIs) and then uses RoIAlign layer to extract features for each RoI. Conventionally, the features are then used for predicting bounding box offsets and classes. Our wrapper architecture still predicts bounding box offsets but it predicts an embedding vector k_i instead of a class for region i. We replace the classification layer with a fully-connected layer with 256 channels and the class-specific bounding box regressor with a class-agnostic one.

We then measure the similarities between a region and all objects by calculating dot products between an embedding vector of a region and all embedding vectors in the memory bank. We apply a contrastive loss function:

$$\mathfrak{L}_{\text{con}} = -\sum_{i=1}^{M} \log \frac{\exp\left(k_i \cdot q_{c_i}/\tau\right)}{\sum_{j=1}^{N} \exp\left(k_i \cdot q_j/\tau\right)} \tag{1}$$

where M is the number of regions, N is the number of objects, c_i is the object for region i and τ is a temperature hyper-parameter to train the detectors to predict similar embedding vectors for the same object. We follow MoCov2 (Chen et al., 2020d) to set τ to be 0.2 for all of our experiments. This loss is only applied to the foreground regions.

We use SGD to optimize the full training loss:

$$\mathfrak{L} = \mathfrak{L}_{\text{con}} + \mathfrak{L}_{\text{reg}} + \mathfrak{L}_{\text{rpn}} \tag{2}$$

where $\mathfrak{L}_{\text{reg}}$ is the bounding box regression loss and $\mathfrak{L}_{\text{rpn}}$ is the loss for RPN. And we use momentum (He et al., 2020; Chen et al., 2020d) to update the parameters in Faster R-CNN for query images with a momentum coefficient of 0.999 and the gradients to update the parameters in Faster R-CNN for target images.

4 Experiments

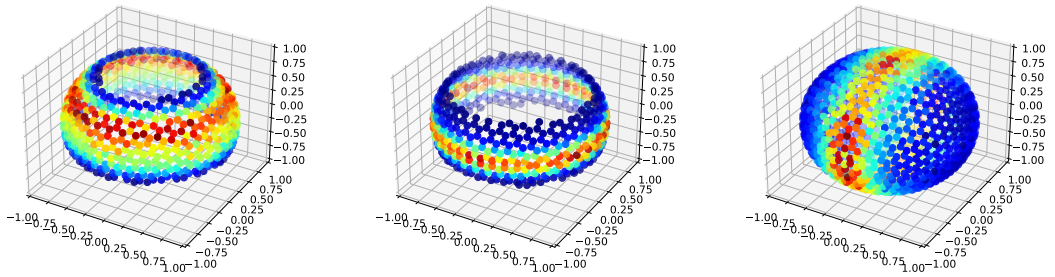
Implementation Details We use 3D models from ShapeNet (Chang et al., 2015), which can be used for non-commercial research, and indoor scenes from SceneNet (Handa et al., 2015), which is released under creative commons license, to construct our datasets. In our ablation studies, we use a subset of ShapeNet models that are used by SceneNet RGB-D (McCormac et al., 2017) and render images at 320×240. In experiments where we compare with existing approaches, we use all ShapeNet models and render images at 640×480.

We use Blender 2.92 [1], an open source 3D computer graphics software, and BlenderProc (Denninger et al., 2019), a Blender library, to render images and generate bounding box and mask annotations.

[1] https://www.blender.org

Table 1: We render multiple datasets of one million images each to demonstrate how pretraining data affects the performance of a detector. We also include SceneNet RGB-D for comparison. Fig. 4 shows the viewpoint distributions of `Random Placement`, `Occlusion` and `Rotation`.

Dataset	Obj Count	Occlu.	Scale Dist. (s / m / l)	Rotation Axes	Scene	COCO AP
`SceneNet RGB-D` (McCormac et al., 2017)	5.41	-	45% / 40% / 15%	-	SceneNet	36.6%
`Random Placement`	8.73	19%	18% / 52% / 30%	Z axis	White Cube	37.2%
`Occlusion`	7.73	32%	23% / 48% / 29%	Z axis	White Cube	37.2%
`Scale Distribution`	8.47	33%	32% / 37% / 31%	Z axis	White Cube	37.5%
`Rotation`	8.10	33%	35% / 36% / 29%	All axes	White Cube	37.7%
`SceneNet Background`	8.72	37%	38% / 34% / 28%	All axes	SceneNet	38.8%
`More Objects`	13.72	38%	33% / 39% / 28%	All axes	SceneNet	39.0%



(a) `Random Placement` (b) `Occlusion` (c) `Rotation`

Figure 4: Viewpoint distribution changes with both the camera and object poses. In `Random Placement` and `Occlusion`, we only rotate along the z-axis so the top and bottom viewpoints are not well covered. In `Occlusion`, we restrict the camera poses so that it is easier to create occlusions between objects in the final images, which reduces the variations in viewpoints. In `Rotation`, we still restrict the camera poses but we rotate the objects along more axes to cover more viewpoints.

We use the Cycles engine from Blender to render all images on GPUs. And we render one million images for each of the dataset in our experiments.

We implement our approach in Detectron2 (Wu et al., 2019) and use the default hyper-parameters, except the training schedule and input resolution, to pretrain our detectors. After pretraining, we initialize a downstream detector with weights in the detector for the target image excluding the classifier and regressor. To provide a fair comparison with other works, we fine-tune the detectors with the same schedule in each downstream setting, which will be described in detail in later sections.

In our ablation experiments, during pretraining, we reduce the input resolution to half of the default input resolution. Our wrapper architecture is pretrained on 4 RTX3090 GPUs for 450k iterations with a batch size of 128 query images and a learning rate of 0.02. Each GPU samples 128 query images with a resolution of 112×112. When we compare our results with other approaches, we use the default input resolution in Detectron2 and increase the resolution of query images to 224×224. We optimize the learning rate schedule on COCO `val2017` under the 1x fine-tuning schedule. The wrapper architecture is pretrained on 8 A6000 GPUs for 750k iterations with an initial learning rate of 0.1 and a cosine annealing schedule (Loshchilov & Hutter, 2017). The target network uses batch normalization while the query network uses exponential moving average normalization (Cai et al., 2021), a variant of batch normalization designed for self-supervised learning.

Synthetic Optimized Layout We optimize the parameters of our rendering pipeline against a set of proxy metrics so that we maximize the effectiveness of the synthetic data. We render multiple sets of synthetic images with different proxy metrics and parameters. For each dataset, we pretrain a Faster R-CNN (Ren et al., 2017) with a FPN (Lin et al., 2017) and ResNet-50 (He et al., 2016), fine-tune it on COCO `train2017` under 1x schedule and evaluate it on COCO `val2017`. Tab. 1 shows the proxy metrics of each dataset and the corresponding validation performance. We also pretrain with SceneNet RGB-D (McCormac et al., 2017), which is a large scale synthetic dataset and consists of 5 million images, in Tab. 1 for comparison.

`Random Placement:` We start with randomly placing objects inside of a textureless cube. Each object is randomly scaled between 0.4 and 2.0 and rotated along the z-axis. The object is randomly placed

Table 2: We compare our pretraining approach with existing pretraining approaches by fine-tuning a Mask R-CNN (`2fc`) with FPN and R-50 on COCO `train2017` under the standard 1x and 2x schedule, and evaluating it on COCO `testdev`. For each existing approach, we fine-tune a Mask R-CNN using the provided pretrained weight to get the test AP. We also include both the validation APs reported by each approach and validation APs reproduced by ourselves.

	1x Schedule						**2x Schedule**					
	AP^{bb}			AP^{mk}			AP^{bb}			AP^{mk}		
	reported `val`	reprod. `val`	`test`	reported `val`	reprod. `val`	`test`	reported `val`	reprod. `val`	`test`	reported `val`	reprod. `val`	`test`
Image Cls.	38.9	-	-	35.4	-	-	40.6	-	-	36.8	-	-
MoCov2 (Chen et al., 2020d)	38.9	39.6	39.9	35.4	35.9	36.2	40.9	41.6	41.8	37.0	37.6	37.9
SwAV (Caron et al., 2020)	-	40.0	40.5	-	36.6	37.2	-	42.1	42.5	-	38.2	38.7
DetCo (Xie et al., 2021)	40.1	40.0	40.3	36.4	36.2	36.6	-	41.5	42.0	-	37.6	38.1
ReSim-FPN^T (Xiao et al., 2021)	40.3	40.4	40.6	36.4	36.6	36.8	41.9	41.8	42.4	37.9	37.9	38.4
MoCov3 (Chen et al., 2021)	-	40.7	41.1	-	37.0	37.5	-	42.2	42.6	-	38.4	38.6
DetCon[2] (Hénaff et al., 2021)	42.7	41.5	41.8	38.2	37.6	37.9	43.4	42.5	42.8	38.7	38.4	38.7
SCRL (Roh et al., 2021)	41.3	41.6	42.0	37.7	37.5	38.0	-	42.8	43.2	-	38.7	39.0
InsLoc (Yang et al., 2021)	42.0	-	-	37.6	-	-	43.3	-	-	38.8	-	-
Ours	-	41.4	41.5	-	37.3	37.5	-	42.1	42.8	-	38.0	38.6

in a location such that it does not collide with other objects and is on the floor. We randomly place a camera at a height between 0.1m and 5.0m, and point it toward a random point at a height between 0m and 2.0m in the central area of the background. This gives us mostly the eye-level and high-angle shots. The Faster R-CNN pretrained with this dataset achieves an AP of 37.2% on COCO `val2017`.
`Occlusion:` Instead of random placement, the object is placed in front of or behind other objects with respect to the camera. The camera pose is adjusted to mostly eye-level shots to create occlusions between objects in the image. This increases the occlusion from 19% to 32% but reduces the viewpoint variation due to more constrained camera placement, as shown in Fig. 4. The AP stays at 37.2%. Later we show that increasing viewpoint variation improves the performance.
`Scale Distribution:` In the previous two configurations, an object is randomly scaled between 0.4 and 2.0 and the majority of the objects in the images are medium size. We divide the range into three intervals, $[0.1, 1.0]$, $[1.0, 2.0]$ and $[2.0, 3.0]$, and randomly select an interval with a probability of 0.7, 0.1 and 0.2 respectively. This adds more small objects to the final dataset and improves the AP from 37.2% to 37.5%.
`Rotation:` In this configuration, we not only rotate an object along the z axis but also the x and y axes. This includes more viewpoints of the objects in our dataset even if we are mostly using eye-level shots. Fig. 4 shows how the viewpoint distributions change between `Scale Distribution` and `Rotation`. This dataset improves the AP from 37.5% to 37.7%, which also explains why there is no improvement in `Occlusion`.
`SceneNet Background:` We use backgrounds from SceneNet and the AP improves to 38.8%.
`More Objects:` We put more objects into the scene by allowing the objects to be floating. This increases the object count per image from 8.72 to 13.72 and the AP from 38.8% to 39.0%.

The above experiments show that our proxy metrics are good indicators of validation performance. When comparing with existing pretraining approaches later, we build upon the `More Objects` configuration, using all 52k models from ShapeNet and rendering images at 640×480 instead of 320×240.

Instance Detection versus Alternative Pretraining Tasks We evaluate the effectiveness of our label-free instance detection pretraining task by comparing it against alternative ways of pretraining including one that uses semantic labels. Using the `SceneNet Background` dataset, we compare our pretraining against two baseline methods to train the classifiers in the detector: (1) we treat each 3D model as an independent class and we have 21k classes in total; (2) we use the semantic labels in ShapeNet and group the 3D models into 148 categories. With the first approach, the training was unstable and diverged. With the second approach achieves an AP of 38.1%. In comparison, the network pretrained on our semantics-free instance detection task achieves an better AP of 38.8%.

Comparisons with Existing Pretraining Approaches To compare our approach with other pretraining approaches, we pretrain a Mask R-CNN (`2fc`) with an FPN and a ResNet-50 on our synthetic data. Following (He et al., 2020), we then fine-tune it on COCO `train2017` under the standard 1x and 2x schedule and the same fine-tuning settings. In Tab. 2, in addition to the validation performance, we also include the test performance for a fair comparison as our learning schedule is optimized on the validation set under the 1x schedule. Since prior work only report validation performance, we fine-tune the network with the provided pretrained weights by ourselves to get the test performance.

[2]DetCon originally uses the TPU implementation of Mask R-CNN instead of Detectron2 and different data augmentation during fine-tuning.

Table 3: We pretrain and fine-tune a Mask R-CNN (`4conv1fc`) with FPN and ResNet-50 by following the 400 epochs training schedule from the train-from-scratch baselines in Detectron2. We pretrain the Mask R-CNN on our pretraining task for the first half of the training schedule.

pretrain	finetune	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
supervised	3x schedule	41.9	61.7	45.8	37.8	59.2	40.5
no	9x schedule	43.6	63.6	47.6	39.3	60.9	42.3
no	100 epochs + LSJ (Ghiasi et al., 2021)	44.7	65.0	49.0	40.3	62.1	43.7
no	200 epochs + LSJ	46.3	66.7	50.7	41.7	64.1	45.0
no	400 epochs + LSJ	47.4	67.6	52.4	42.5	65.2	46.1
Ours	200 epochs + LSJ	47.8	68.2	52.7	43.0	65.7	46.6

Table 4: We fine-tune a Faster R-CNN with only 10% of COCO `train2017`.

pretrain	AP	AP^{50}	AP^{75}
random	17.8	32.0	17.9
ImageNet	22.6	38.4	23.5
MoCov2	20.9	34.8	21.7
SimCLRv2 (Chen et al., 2020c)	22.1	37.3	23.0
SwAV	25.5	43.3	26.4
BYOL (Grill et al., 2020)	25.5	42.3	26.9
SCRL	26.4	43.2	28.0
SCRL (reprod.)	26.7	43.3	28.3
Ours	26.3	41.3	28.2

Table 5: We evaluate on the few-shot learning task.

	10 Shot					
	Base			Novel		
	AP	AP50	AP75	AP	AP50	AP75
ImageNet	32.9	52.2	36.3	8.8	17.4	7.9
DetCo	34.4	54.2	37.8	9.5	18.0	8.9
MoCov3	35.9	56.6	39.7	8.7	16.9	8.2
ReSim-FPN^T	34.7	54.2	38.5	9.4	17.4	9.0
SCRL	36.0	55.9	39.5	9.9	18.8	9.4
SwAV	36.0	57.0	39.6	10.1	19.4	9.5
Ours	35.9	55.0	39.7	9.4	17.3	8.9

Our reproduced APs are similar to or better than the reported APs except for DetCon which originally uses the TPU implementation of Mask R-CNN and different data augmentations during fine-tuning. For the network pretrained on ImageNet, the TPU version achieves an AP of 39.6% while the Detectron2 version achieves an AP of 38.9%. InsLoc modifies the architecture of Mask R-CNN by adding four convolution layers to the bounding box head, while other approaches use the architecture of Mask R-CNN in MoCo. We include the reported number (which is not directly comparable to other approaches) in Tab. 2 for reference. Our approach achieves results competitive to the state-of-the-art pretraining approaches such as MoCov3, SCRL and DetCon. It is worth noting that our approach uses only 8 A6000 GPUs for pretraining while MoCov3 uses 16 V100 32G GPUs, SCRL uses 32 V100 GPUs and DetCon uses 128 TPU v3 workers.

Pretraining versus Train-from-scratch He et al. (He et al., 2019) show that a detector trained from scratch can be a strong baseline if it is trained long enough. Detectron2 provides a train-from-scratch baseline where it trains a Mask R-CNN (`4conv1fc`) with an FPN and a ResNet-50 and strong data augmentation (Ghiasi et al., 2021) for 400 epochs on COCO from scratch. This baseline achieves an AP of 47.4%, while the baseline pretrained on ImageNet only achieves an AP of 41.9%. To verify that our pretraining still helps under a long fine-tuning schedule, we follow the 400-epochs training schedule where we use the first half for pretraining and fine-tune our network for 200 epochs. We use the same data augmentation and batch size. Tab. 3 shows that our approach outperforms the strong train-from-scratch baselines.

Low Data Regime Following SCRL (Roh et al., 2021), we evaluate our approach in low data regime by fine-tuning the detector with only 10% of COCO `train2017` data, as shown in Tab. 4 which is adapted from SCRL. We also use the two-stage fine-tuning approach (TFA) from FsDet (Wang et al., 2020b) to evaluate our approach on the few-shot learning task, as shown in Tab. 5. The 20 classes that are in both COCO and PASCAL VOC are used as novel classes while the 60 classes that are only in COCO are used as base classes. Each novel class has 10 training examples. TFA first fine-tunes the whole network on the base classes and then fine-tunes only the classifiers on the novel classes. Experiments show that our approach achieves results competitive to the state-of-the-art approaches.

Limitations Our "SOLID" approach mainly focuses on the spatial layout of the objects when generating synthetic images. But there are other aspects in the rendering pipeline that can potentially generate more effective pre-training data. Tab. 1 shows that using backgrounds from SceneNet outperforms a textureless cube so it is possible that even more diversified backgrounds would be beneficial. Generating more photo realistic synthetic images may also reduce the domain gap between real and synthetic images.

Conclusion We have introduced SOLID, a new approach to pretraining an object detector with synthetic data. Experiments on COCO show that synthetic data can be highly effective data for pretraining object detectors.

Reproducibility The code for reproducing the results is included as supplementary materials. It includes instructions to render the final dataset, pretrain and fine-tune a model, and links to necessary data for rendering and a pretrained model which are stored in an anonymous Google account.

REFERENCES

Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars M. Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning. In *BMVC*. BMVA Press, 2017. 1, 4

Phil Ammirato, Cheng-Yang Fu, Mykhailo Shvets, Jana Kosecka, and Alexander C. Berg. Target driven instance detection. *CoRR*, abs/1803.04610, 2018. 2, 3

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*. OpenReview.net, 2020. 1

Hangbo Bao, Li Dong, and Furu Wei. Beit: BERT pre-training of image transformers. *CoRR*, abs/2106.08254, 2021. 1

Amir Bar, Xin Wang, Vadim Kantorov, Colorado J. Reed, Roei Herzig, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson. Detreg: Unsupervised pretraining with region priors for object detection. *CoRR*, abs/2106.04550, 2021. 1, 3

Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007. 3

Erik Bochinski, Volker Eiselein, and Thomas Sikora. Training a convolutional neural network for multi-class object detection using solely virtual world data. In *AVSS*, pp. 278–285. IEEE Computer Society, 2016. 4

Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV (6)*, volume 7577 of *Lecture Notes in Computer Science*, pp. 611–625. Springer, 2012. 1

Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *CVPR*, pp. 194–203. Computer Vision Foundation / IEEE, 2021. 7

Yue Cao, Zhenda Xie, Bin Liu, Yutong Lin, Zheng Zhang, and Han Hu. Parametric instance classification for unsupervised visual feature learning. In *NeurIPS*, 2020. 1

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV (14)*, volume 11218 of *Lecture Notes in Computer Science*, pp. 139–156. Springer, 2018. 1

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 1, 3, 8

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 3, 4, 6

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1691–1703. PMLR, 2020a. 1

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020b. 1, 3

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020c. 3, 9

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, pp. 15750–15758. Computer Vision Foundation / IEEE, 2021. 1

Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020d. 1, 3, 6, 8

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, pp. 9620–9629. IEEE, 2021. 3, 8

Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. UP-DETR: unsupervised pre-training for object detection with transformers. In *CVPR*, pp. 1601–1610. Computer Vision Foundation / IEEE, 2021. 3

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. IEEE Computer Society, 2009. 3

Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *CoRR*, abs/1911.01911, 2019. 6

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *ICCV*, pp. 2070–2079. IEEE Computer Society, 2017. 1

Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, pp. 2918–2928. Computer Vision Foundation / IEEE, 2021. 9

Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pp. 580–587. IEEE Computer Society, 2014. 3

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020. 9

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR (2)*, pp. 1735–1742. IEEE Computer Society, 2006. 3

Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *CoRR*, abs/1511.07041, 2015. 3, 4, 6

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016. 7

Kaiming He, Ross B. Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, pp. 4917–4926. IEEE, 2019. 3, 9

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9726–9735. Computer Vision Foundation / IEEE, 2020. 1, 3, 6, 8

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. 1

Olivier J. Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aäron van den Oord, Oriol Vinyals, and João Carreira. Efficient visual pretraining with contrastive detection. In *ICCV*, pp. 10066–10076. IEEE, 2021. 1, 3, 8

Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *ECCV Workshops (1)*, volume 11129 of *Lecture Notes in Computer Science*, pp. 682–697. Springer, 2018. 4

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006. 3

Jiabo Huang, Qi Dong, Shaogang Gong, and Xiatian Zhu. Unsupervised deep learning by neighbourhood discovery. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2849–2858. PMLR, 2019. 1

Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *ICRA*, pp. 746–753. IEEE, 2017. 1

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, pp. 1920–1929. Computer Vision Foundation / IEEE, 2019. 1

Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *ECCV (17)*, volume 12362 of *Lecture Notes in Computer Science*, pp. 574–591. Springer, 2020. 1

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV (5)*, volume 8693 of *Lecture Notes in Computer Science*, pp. 740–755. Springer, 2014. 2, 3

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pp. 936–944. IEEE Computer Society, 2017. 7

Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *3DV*, pp. 218–227. IEEE, 2021. 1

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR (Poster)*. OpenReview.net, 2017. 7

Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pp. 4040–4048. IEEE Computer Society, 2016. 1

John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. Scenenet RGB-D: can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *ICCV*, pp. 2697–2706. IEEE Computer Society, 2017. 6, 7

Jean-Philippe Mercier, Mathieu Garon, Philippe Giguère, and Jean-François Lalonde. Deep template-based object instance detection. In *WACV*, pp. 1506–1515. IEEE, 2021. 2, 3

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, pp. 6706–6716. Computer Vision Foundation / IEEE, 2020. 1

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, pp. 2536–2544. IEEE Computer Society, 2016. 1

Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *ICCV*, pp. 1278–1286. IEEE Computer Society, 2015. 4

Aayush Prakash, Shoubhik Debnath, Jean-Francois Lafleche, Eric Cameracci, Gavriel State, Stan Birchfield, and Marc T. Law. Self-supervised real-to-sim scene generation. In *ICCV*, pp. 16024–16034. IEEE, 2021. 1

Marc'Aurelio Ranzato, Christopher S. Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS*, pp. 1137–1144. MIT Press, 2006. 3

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6): 1137–1149, 2017. 3, 7

Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV (2)*, volume 9906 of *Lecture Notes in Computer Science*, pp. 102–118. Springer, 2016. 4

Byungseok Roh, Wuhyun Shin, Ildoo Kim, and Sungwoong Kim. Spatially consistent representation learning. In *CVPR*, pp. 1144–1153. Computer Vision Foundation / IEEE, 2021. 1, 3, 8, 9

Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, pp. 3626–3633. IEEE Computer Society, 2013. 3

Alireza Shafaei, James J. Little, and Mark Schmidt. Play and learn: Using video games to train computer vision models. In *BMVC*. BMVA Press, 2016. 4

Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: end-to-end object detection with learnable proposals. In *CVPR*, pp. 14454–14463. Computer Vision Foundation / IEEE, 2021. 3

Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, pp. 10778–10787. Computer Vision Foundation / IEEE, 2020. 3

Zachary Teed and Jia Deng. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. In *NeurIPS*, pp. 16558–16569, 2021. 1

Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *CVPR Workshops*, pp. 969–977. Computer Vision Foundation / IEEE Computer Society, 2018a. 1, 4

Jonathan Tremblay, Thang To, and Stan Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. In *CVPR Workshops*, pp. 2038–2041. Computer Vision Foundation / IEEE Computer Society, 2018b. 1

Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective search for object recognition. *Int. J. Comput. Vis.*, 104(2):154–171, 2013. 3

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pp. 1096–1103. ACM, 2008. 1

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010. 1

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian A. Scherer. Tartanair: A dataset to push the limits of visual SLAM. In *IROS*, pp. 4909–4916. IEEE, 2020a. 1

Xin Wang, Thomas E. Huang, Joseph Gonzalez, Trevor Darrell, and Fisher Yu. Frustratingly simple few-shot object detection. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9919–9928. PMLR, 2020b. 9

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 7

Tete Xiao, Colorado J. Reed, Xiaolong Wang, Kurt Keutzer, and Trevor Darrell. Region similarity representation learning. In *ICCV*, pp. 10519–10528. IEEE, 2021. 1, 3, 8

Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *ICCV*, pp. 8372–8381. IEEE, 2021. 3, 8

Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining. In *CVPR*, pp. 3987–3996. Computer Vision Foundation / IEEE, 2021. 3, 8

Mang Ye, Xu Zhang, Pong C. Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, pp. 6210–6219. Computer Vision Foundation / IEEE, 2019. 1

Feihu Zhang, Victor Adrian Prisacariu, Ruigang Yang, and Philip H. S. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, pp. 185–194. Computer Vision Foundation / IEEE, 2019. 1

Feihu Zhang, Xiaojuan Qi, Ruigang Yang, Victor Prisacariu, Benjamin W. Wah, and Philip H. S. Torr. Domain-invariant stereo matching networks. In *ECCV (2)*, volume 12347 of *Lecture Notes in Computer Science*, pp. 420–439. Springer, 2020. 1

Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. *CoRR*, abs/2103.07461, 2021. 3