Creation of the 2D synthetic dataset and Transfer learning for Custom object detection

Gowtham Buvalli Chikkathammaiah
Hochschule Hof, Hof, Bayern
gowtham.buvalli.chikkathammaiah@hof-university.com

Abstract

Since there is a recent increase in demand of the Deep Neural Networks field application especially the computer vision field in particular the sub-topic object detection is finding a lot of applications in the industry such as smart fridges, robot arm grasping, etc. Even though there are plenty of available open-source datasets, there are still a few very ways of creating the custom datasets required for the off-shelf custom object product detections. These are labor-intensive, time-consuming, and quite inefficient to solve the given problem. The Deep neural networks training on them from scratch are slow which makes practically impossible for the production release in industrial applications. The proposed research tries to address these particular problems by providing the solution to create the custom datasets efficiently. Upon these created datasets, a deep neural network computer vision model is transfer learning trained and compared with its peers in terms of performance relative to the industry standards. The important criteria affecting the synthetic data generation and its effect on the trained model will be discussed in detail in the paper.

Keywords

Synthetic data generation, Computer vision, Object detection

1. Introduction

The capability of detecting objects in challenging environments is fundamental for many machine vision and robotics tasks. Recently, proposed modern deep convolutional architecture such as Faster R-CNNs [24], SSD [16], R-FCN [5], Yolo9000 [23], and RetinaNet [15] have achieved very impressive results. However, the training of such models with millions of parameters requires a massive amount of labeled supervised training data to achieve state-of-the-art results. Clearly, the creation of such massive datasets has become one of the main limitations of these approaches: they require human input, are very costly, time-consuming, and error-prone.

Training with synthetic data is very attractive because it decreases the burden of data collection and annotation. Theoretically, this enables generating an infinite amount of training images with large variations, where labels come at no cost. In addition, training with synthetic samples allows us to precisely control the rendering process of the images and thereby the various properties of the dataset. However, the main challenge for successfully applying such approaches in practice still remains, i.e. how to bridge the so-called “domain gap” between synthesized and real images. As observed in [30], methods trained on synthetic data and evaluated on real data usually result in deteriorated performance.

Figure 1. Example results of Faster R-CNN [24] trained on purely synthetic data from 3D models. In this paper, we introduce a novel approach for creating synthetic training data for object detection that generalizes well to real data with least time. Our trained model is able to robustly detect objects under various poses, partial occlusion, and illumination changes.

To address this challenge, several approaches have focused on improving the realism of training data [9, 1, 8, 33], mixing synthetic and real data [6, 8, 21], leveraging architectures with frozen pre-trained feature extractors [10, 14, 22], or using domain adaptation or transfer learning as in [26, 4, 7].

“Domain Randomization” as introduced in [30] is another strategy to narrow the gap between real and synthetic data. The authors hypothesized that high randomization of the synthesis process yields better generalization as reality is seen by the trained models as a mere instance of the larger domain space it was trained on. They showed promising first results with a few objects in simple scenarios. More recently, this idea was extended with the addition of real background images mixed with partial domain randomized scenes [31, 20], and further improved through photo-realistic rendering [32].

While those approaches provided impressive results, the main drawback still remains i.e. their dependence on real data.

In this paper, we introduce a novel way to create purely synthetic training data for object detection. We leverage a large dataset of 3D foreground models which we densely render in a full domain randomized fashion to create our dataset images. We are able to generate a locally realistic plain background which makes our trained models robust to environmental changes. On top of these background images, the 3D objects of interest are rendered. During training, the data generation process follows a curriculum strategy that ensures that all foreground models are presented to the network equally under all possible poses with increasing complexity. Finally, we add randomized illumination, blur, and noise.

Our approach doesn't require complex scene compositions as in [32, 9, 1, 8, 33], difficult photo-realistic image generation as in [32, 9, 1], or real background images to provide the necessary background clutter [10, 14, 22, 31, 20, 32], and scales very well to a large number of objects and general detection capabilities.

To the best of our knowledge, "The first to present such a purely synthetic method for generating training data for object instance detection that outperforms models trained on real data" [34]. This research forms as a baseline for our work. Furthermore, The demonstration of experiment of the benefits of curriculum strategy versus random pose generation. The proposed work also shows that generated images should ideally be composed of synthetic content only and that the whole background image should be filled with plain background. Finally, we perform thorough ablation experiments to highlight the contributions of the different components of our pipeline.

In the remainder of the paper, The discussion of the related work, description of the pipeline for generating synthetic images, demonstrating the variability of the elements with respect to synthetic data, and detail the conclusions and future works.

2. Related Work

A common approach to **improve detection performance is to extend a real training dataset by adding synthetic data**. For instance, [28, 6, 8] train a single network on such a mixed dataset. While these methods demonstrate a significant improvement over using real data only, they still require at minimum real domain-specific background images as in [28].

[6, 8] follow an image composition approach **to create synthetic images by combining cut out objects from different images**. These approaches have the benefit of using data from the same domain, as the cut out objects are copies of real images, and as such, they closely match the characteristics of the real world. The main limitation of these approaches is that they require performing the cumbersome process of capturing images of the objects from all possible viewpoints and mask them. In particular, these methods can't produce images from different views or different lighting conditions once the object training set is fixed. This is a clear limitation.

Other lines of work **utilize photo-realistic rendering and realistic scene compositions to overcome the domain gap by synthesizing images that match the real world as close as possible** [9, 13, 25, 17, 1, 8, 33, 18]. While these methods have shown promising results they face many hard challenges. First, producing photo-realistic training images requires sophisticated rendering pipelines and considerable CPU/GPU resources. Second, realistic scene composition is a hard problem on its own usually done by hand. Third, modern rendering engines used for creating synthetic scenes heavily take advantage of the human perception system to fool the human eye. However, these tricks do not necessarily work on neural networks and thus require more effort to bridge the domain gap.

Following their success for image generation, **Generative Adversarial Networks** (GANs) have been used in [27, 3] to further bridge the domain gap. However, such approaches bring substantial additional complexity as they are difficult to design and train. To the best of our knowledge, they have not been applied to detection tasks as of now.

Another line of work **utilizes domain adaptation or transfer learning [26, 4, 7, 12] to bridge the domain gap between the synthetic and real domain**. This can be achieved by coupling two predictors, one for each domain, or by combining the data from two domains. Domain adaptation and transfer learning have applications far beyond the transfer from synthetic to real data. Still, they require a significant amount of real data.

Our method falls into the category of d**omain randomization** [30, 31, 32, 20, 2]. The basic idea is to alter the simulated data with non-realistic changes so that reality seems to be just a variation. [30] introduced the concept of domain randomization to overcome the domain gap. They use non-realistic textures for rendering synthetic scenes to train an object detector that generalizes to the real world.

In another line of work, [32] combines **domain randomization and photo-realistic rendering**. They generate two types of data: First, synthetic images with random distractors and variations that appear unnatural with real photographs as background as introduced in [31], and second, photorealistic renderings of randomly generated scenes using a physics engine to ensure physical plausibility. The combination of these two types of data yields great improvement over only one source of data and allows the network to generalize to unseen environments. [20] uses structured domain randomization, which allows the network to take context into account.

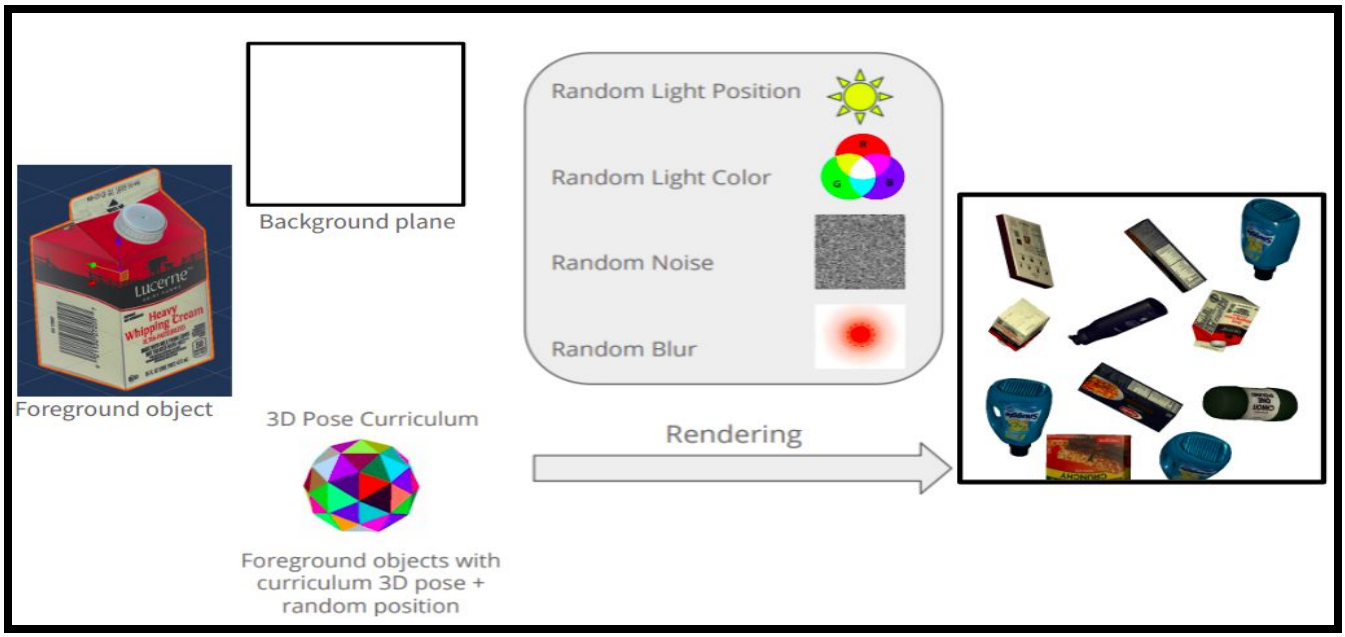


Figure 2*. Our synthetic data generation pipeline. For each training image, we generate a background scene until each pixel in the resulting image would be covered (see Section 3.1). Then, we add one or many foreground objects to the scene; each object is randomly positioned in the image but follows a deterministic schedule for rotation and scale (see curriculum strategy in Section 3.2). Finally, we render the scene using simple Phong illumination [19] with a randomly placed light source with a random light color, followed by adding random noise to the image and random blur. We also compute a tightly fitting bounding box using the object's 3D model and the corresponding pose.*

3.Synthetic data generation Method

In this section, we present our pipeline for generating synthetic training data as shown in Fig. 2. As opposed to previous methods [6, 8, 21], we do not try to diminish the domain gap by mixing synthetic and real images but create purely synthesized training samples. Each training sample is generated by blending two image layers - a purely synthetic background layer, a foreground object layer built following a curriculum strategy. Since we are dealing with object instance detection and are interested in rendering our objects geometrically correct, we make use of the internal camera parameters, i.e. focal length and principal point. To gain additional robustness, we allow for slight random variations of these parameters during training. In the remainder of this section, we will describe in detail how we create each of these layers and the underlying principles which guided the design of the rendering pipeline.

3.1. Background Layer

The background layer is a white plain background screen generated in unity 3D. As required, other realistic backgrounds can be used. For our use case, the white ground serves the requirement.

3.2. Foreground objects

For each foreground object, we start by generating a large set of poses uniformly covering the pose space in which we want to be able to detect the corresponding object. To do so, we use the approach described in [10] and generate rotations by recursively dividing an icosahedron, the largest convex regular polyhedron. This approach yields uniformly distributed vertices on a sphere and each vertex represents a distinct view of an object defined by two out-of-plane rotations. In addition to these two out-of-plane rotations, we also use equally sampled in-plane rotations. Furthermore, we sample the distance at which we render a foreground object inversely proportional to its projected size to guarantee an approximately linear change in pixel coverage of the projected object between consecutive scale levels. Opposite to the plain background, we render the foreground objects based on a curriculum strategy (see Fig. 3). This means that there is a deterministic schedule at which step each object and pose should be rendered:

1. We start with the scale that is closest to the camera and gradually move to the one that is farthest away. As a result, each object initially appears largest in the image, being, therefore, easier to learn for the network. As learning proceeds, the objects become smaller and more difficult for the network to learn.

2. For each scale, we iterate through all possible out-of-plane rotations, and for each out-of-plane rotation, we iterate through all in-plane rotations in the x and y-axis.

3. Once we have a scale, an out-of- and an in-plane rotation, we iterate through all objects and render each of them with the given pose at a random location using a uniform distribution.

4. After having processed all objects, at all in and out-of-plane rotations, we move to the next scale level.

For rendering, we allow cropping of foreground objects at the image boundaries up to 50%. In addition, we allow for overlap between each pair of foreground objects up to 30%. For each object, we randomly try to place it n = 100 times in a foreground scene. If it can't be placed within the scene due to violations of the cropping or overlap constraints we stop processing the current foreground scene and start with the next one. For the subsequent foreground scene, we start where we have left off the last scene.

3.3. Occlusion layer generation

We also generate an occlusion layer where the random objects from the foreground objects partially occlude in between them. This is done by determining the bounding box of each rendered foreground object and by rendering a randomly selected occluding object at a uniform random location within this bounding box. The occluding object is randomly scaled such that its projection covers a certain percentage of the corresponding foreground object (in a range of 10% to 30% of the other object). The pose and color of the occluding object are randomized.

3.4. Postprocessing and Layer Fusion

Having the background, foreground, and occlusion layer, we fuse all three layers into one combined image: the occlusion layer is rendered on top of the foreground layer and the result is rendered on top of the background layer. Furthermore, we add random light sources with random perturbations in the light color. Finally, we add white noise and blur the image with a Gaussian kernel where both, the kernel size and the standard deviation, are randomly selected. Thus, background, foreground, and the occluding parts share the same image properties which are contrary to other approaches [10, 14, 22, 31, 20, 32] where real images and synthetic renderings are mixed. This makes it impossible for the network to differentiate foreground vs. background merely on attributes specific to their domain. In Fig. 2 we show some images generated with our method.

4. Deep Neural Network Model

We report detailed experiments and results underpinning the benefits of our strategy. After describing our experimental setup, we demonstrate that synthetic data generation permits us to train state-of-the-art architectures at no cost that outperforms models trained on real data. Furthermore, we show through ablation experiments the benefits of curriculum vs random pose generation, the effects of the relative scale of foreground objects with respect to background, the effects of the number of foreground objects rendered per image and finally the effects of random colors and blur.

4.1. 3D Models

In all our experiments, we focus on the detection of 10 different instances of foreground objects showing all very different properties in terms of colors, textures (homogeneous color vs. highly textured), 3D shape, and materials (reflective vs. non-reflective). As illustrated by Fig. 3, these objects are mostly classical retail objects that can be found in a supermarket. For each foreground object, we generated a textured 3D model using our in-house 3D scanner.

4.2. Real Training and Evaluation Data

The way of real image dataset obtained is as follows, the collection was performed on all our real data acquisitions using the Intel Realsense D435 camera. While this camera permits the capture of RGB and depth images, the focus was on RGB only. Using the above camera, the training data was built and an evaluation benchmark of 1158 and 250 real RGB images, respectively, at a resolution of 960x720. The benchmark training set consists of images picturing random subsets of the objects of interest disposed of on a cluttered background and in different lighting conditions (natural day/evening light vs. artificial light). The evaluation set consists of images displaying the objects of interest randomly distributed in shelves, boxes, or layed out over random clutter. Since it is crucial for reliable object detection, both sets each object is shown in various poses and appears equally (roughly around 120 times for each object in the training set and around 40 times in the evaluation set). All those images were labeled by human annotators and additionally controlled by another observer to ensure the highest label quality. This step is permitted to correct around 10% of mislabeled examples which is crucial for a fair comparison with synthetic data benefiting from noise-free labels. The amount of time spent for acquiring

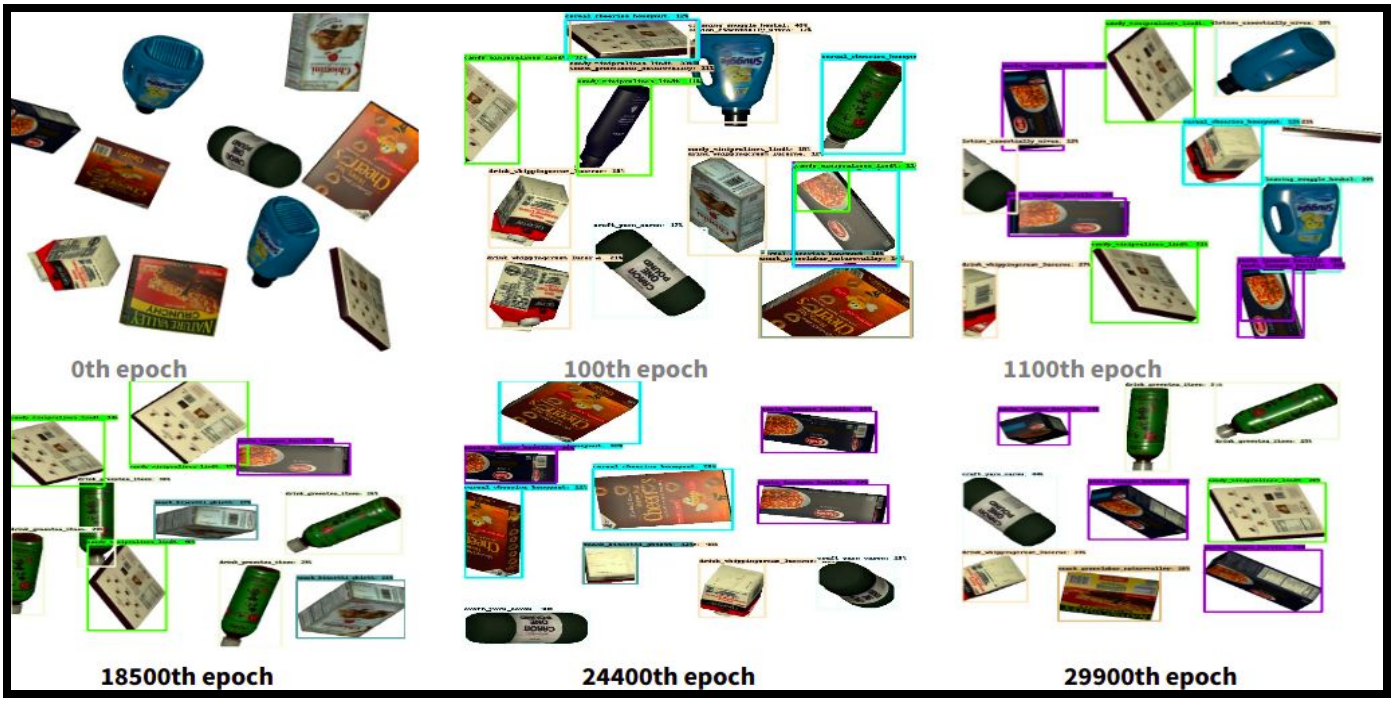


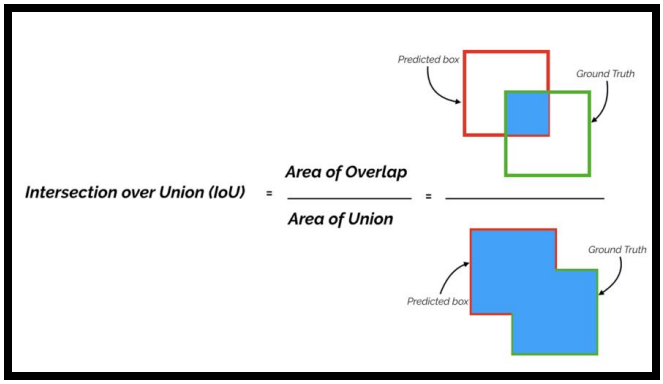
Figure 3: Contains the random selection of inference images and their respective epochs. The Figure explains the training progress epoch vs model accuracy visually. Top left explains that at 0th epoch, the model couldnt recognise any objects but as the training progresses the accuracy and classification loss decreases. The 100th epoch has multiple localizations. The 18,500th epoch has the wrong classification.

the real images was around 10 hours and labeling required approximately 185 hours for the training set, with 6 additional hours spent for correction[34]. Note that for real data, acquisition and annotation efforts are always required if new objects are added to the dataset, and images mixing the new objects and the legacy objects needed are generated. In contrast, time spent scanning the 64 foreground objects was roughly 5 hours, and this is a one-time effort: if new objects are added to the dataset, only one scan per additional object is required. These data are used to train the model from scratch and later on, transfer learning is applied on the model.

4.3. Network Architecture

Modern state-of-the-art object detection models consist of a feature extractor that aims at projecting images from the raw pixel space into a multi-channel feature space and multiple heads that tackle different aspects of the detection problems, such as bounding box regression and classification. In the present work, we use the popular Faster R-CNN [24] architecture with an Inception ResNet 50 feature extractor [29]. Weights of the feature extractor have been pre-trained on the ImageNet dataset. Our implementation uses Google's publicly available open-source implementation of Faster R-CNN [11].

4.3 Evaluation Metric Mean Average Precision



The above formula explains the mean of total number of classes with area overlapped to area of union.

5. Experiments

The Main goal for the proposed research is to generate custom dataset and train them quickly for production deployment.

The proposed system can create 100,000 datasets as per predefined curriculum in a span of 10 minutes, for manual annotations it would have taken 4 months for a team of two.

The proposed system was trained in the transfer learning process for 11,000 epochs at 8 batches to identify all

objects correctly i.e. equivalent less than 15 mins of training. As observed in figure 3, Further training led to decrease in the localization loss but not classification loss.

5.1 Effect of foreground object density in the image

Another crucial factor affecting the dataset creation is the foreground object density. It is observed that there is an inverse proportional correlation between the objects and accuracy i.e. More the object density lessens the accuracy.

5.2 Effect of Foreground object size on image

Object size in individual frames is important as better the object size better the feature extraction. The experiment with lesser object size showed poor results in classification loss. When the object size was increased, the best results were observed. Elimination of depth in the image resulted in better accuracy i.e. Only the variation in the X and Y dimension are considered.

5.3 Effect of background object in the image

The impact of the addition of distraction background objects and colourful background textures are discussed in literature surveys to affect the accuracy of the model positively. As observed during this research, there was not much effect in the localization loss.

5.4 Effect of Brightness, Hue variation on the image

By variation of the different values of Brightness and hue on the synthetic dataset are crucial in object recognition tasks. Better the variation of above factors, better the variety of dataset for training the model better the feature extraction of the model. As all image recognition models learn from RGB of each pixel value, any small variation which is/or non noticeable for human eyes bears a significant impact for feature extraction. In the proposed experiment, the casting of shadows are removed and ambient light is varied in every image to procure a diverse dataset.

The proposed system evaluation metrics for object detection tasks is “Mean Average Precision” [MAP] was at .67 and original paper at .75 IOU [34]. However, considering the size of product portfolio in original paper i.e.60 classes of original paper and 10 classes in proposed research. They cannot be tangibly compared.

6. Results explanation and Comparison

```
Average precision values per class:
candy_minipralines_lindt: 0.649908
cereal_cheerios_honeynut: 0.480704
cleaning_snuggle_henkel: 0.947008
craft_yarn_caron: 0.725478
drink_greentea_itoen: 0.851264
drink_whippingcream_lucerne: 0.860232
lotion_essentially_nivea: 0.645522
pasta_lasagne_barilla: 0.465118
snack_biscotti_ghiott: 0.656190
snack_granolabar_naturevalley: 0.459366
Mean Average Precision @50 is:0.6740790438601992
```

Figure 4: illustrates the object categories with respect to the precision value per class at 50 IOU.

The Figure 4 explains the deep neural networks predictability value for each class i.e. For the product cleaning_snuggle_henkel the CV model can predict the retail product with 94% accuracy whereas for the product snack_granolabar_naturevalley can only be identified 45% successfully. **The overall average i.e. Mean Average Precision at 50 IOU is observed at 67.4%. whereas the state of art as of today, for YOLOv4-p7 is at 74.2%.**

IOU Threshold	Mean Average Precision
25	0.67668
50	0.674079
75	0.66931
90	0.387023

Figure 5 illustrates the IOU threshold against the Mean average precision.

The Intersection Over Union(IOU) is the filtering criteria for the overlapping of model prediction to ground truth values. The 25% explains that if more than 25% is overlapping then predictions are considered true. In above Figure 5, there is less observed difference between 25% and 75% IOU. As the model can rightly predict the objects above 57% overlap area. But with @75 IOU and @90 IOU there is a stark difference observed. This helps in understanding the knowledge learnt by CV models during training.

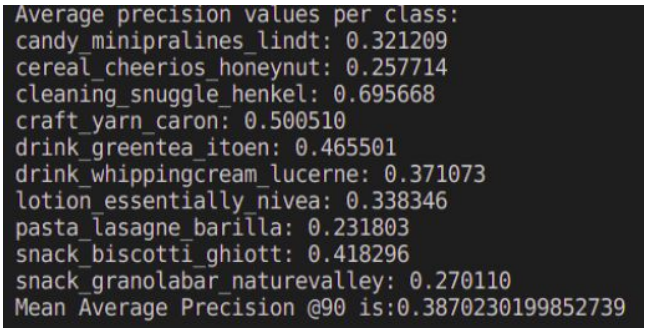


Figure 6 illustrates the object categories with respect to the precision value per class at 90 IOU.

The Figure 6 clearly explains the stark difference observed in the above section, that the model needs more variance in the dataset to extract information for each class i.e. More training rounds with different light intensity variations need to be accommodated in the dataset for better learning curve.

The proposed work not only demonstrated the way of training the model for industrial applications with accuracy and time efficiency but also the process of creating the synthetic dataset at huge scale. The proposed model differed with the state of art method by 10 MAP but for practical industry applications, this system passed the cut. This technology will be applied to production for the computer vision product at Livello technologies, Dusseldorf.

7. Further Improvements

The system proposed has proved enough problem-solving capabilities for industrial applications, but further changes can be experimented for the advancement in research field such as

First, During training the model using transfer learning technique, the tensor consisting of individual product class and its corresponding 2D coordinates maximum repetition can be capped. As this intentional limitation helps in exposure to a variety of samples during training the model.

Secondly, The proposed solution has 10 different products. the product portfolio range should be enhanced and tested for the results.

Third, The other deep neural network architectures such as Single-shot detection using Mobilnet SSD or You Only Look Once (YOLO) have performed better in terms of accuracy and speed in the object detection space. These architectures should also be benchmarked against the proposed solution.

Fourth, The dataset created in this procedure doesn't have any background distraction objects. By including them in the data set creation pipeline, better accuracy of object prediction may be observed.

Although, above mentioned improvements are from our keen observation during research, only the through experimentation would yield the results.

8. References

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes. In British Machine Vision Conference, 2017. 1, 2

[2] J. Borrego, A. Dehban, R. Figueiredo, P. Moreno, A. Bernardino, and J. Santos-Victor. Applying Domain Randomization to Synthetic Data for Object Category Detection. ArXiv e-prints, July 2018. 2

[3] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level Domain Adaptation with Generative Adversarial Networks. In Conference on Computer Vision and Pattern Recognition, 2017. 2

[4] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain Separation Networks. In Advances in Neural Information Processing Systems, 2016. 2

[5] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object Detection via Region-Based Fully Convolutional Networks. In Advances in Neural Information Processing Systems, 2016. 1

[6] D. Dwivedi, I. Misra, and M. Hebert. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. In arXiv Preprint, 2017. 2, 3

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain Adversarial Training of Neural Networks. In Journal of Machine Learning Research, 2016. 2

[8] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka. Synthesizing Training Data for Object Detection in Indoor Scenes. In Robotics: Science and Systems Conference, 2017. 1, 2, 3

[9] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic Data for Text Localisation in Natural Images. In Conference on Computer Vision and Pattern Recognition, 2016. 1, 2

[10] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In Proceedings of the ECCV Workshop on Recovering 6D Object Pose, 2018. 2, 3, 5, 7

[11] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed and Accuracy Trade-Offs for Modern Convolutional Object Detectors. In Conference on Computer Vision and Pattern Recognition, 2017. 5

[12] T. Inoue, S. Chaudhury, G. De Magistris, and S. Dasgupta. Transfer Learning From Synthetic To Real Images Using Variational Autoencoders For Precise Position Detection. ArXiv e-prints, July 2018. 2

[13] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, and R. Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real-world tasks? CoRR, abs/1610.01983, 2016. 2

[14] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: making RGB-based 3d detection and 6d pose estimation great again. CoRR, abs/1711.10006, 2017. 2, 5

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr. The focal loss for dense object detection (best student paper award). In International Conference on Computer Vision, 2017. 1

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: Single Shot Multibox Detector. In European Conference on Computer Vision, 2016. 1

[17] C. Mitash, K. E. Bekris, and A. Boularias. A Self-Supervised Learning System for Object Detection Using Physics Simulation and Multi-View Pose Estimation. In International Conference on Intelligent Robots and Systems, 2017. 2

[18] Y. Movshovitz-attias, T. Kanade, and Y. Sheikh. How Useful is Photo-Realistic Rendering for Visual Learning? In European Conference on Computer Vision, 2016. 2 [

19] B. T. Phong. Illumination for Computer Generated Pictures. In Communications of the ACM, 1975. 4

[20] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In arXiv, 2018. 2, 3, 5, 8

[21] M. Rad and V. Lepetit. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects Without Using Depth. In International Conference on Computer Vision, 2017. 2, 3

[22] P. S. Rajpura, R. S. Hegde, and H. Bojinov. Object detection using deep CNN trained on synthetic images. In arXiv, 2017. 2, 5

[23] J. Redmon and A. Farhadi. Yolo9000: Better, Faster, Stronger. In Conference on Computer Vision and Pattern Recognition, 2017. 1
[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems. 2015. 1, 5
[25] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for Data: Ground Truth from Computer Games. In European Conference on Computer Vision, 2016. 2
[26] A. Rozantsev, M. Salzmann, and P. Fua. Beyond Sharing Weights for Deep Domain Adaptation. In Conference on Computer Vision and Pattern Recognition, 2017. 2
[27] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. In Conference on Computer Vision and Pattern Recognition, 2017. 2
[28] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In ICCV, 2015. 2
[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-V4, Inception-Resnet and the Impact of Residual Connections on Learning. In American Association for Artificial Intelligence Conference, 2017. 5
[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In International Conference on Intelligent Robots and Systems, 2017. 1, 2
[31] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In Workshop on Autonomous Driving, CVPR-Workshops, 2018. 2, 3, 5, 8
[32] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep objects pose estimation for semantic robotic grasping of household objects. In Conference on Robot Learning (CoRL), 2018. 2, 3, 5
[33] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from Synthetic Humans. In Conference on Computer Vision and Pattern Recognition, 2017. 1, 2
[34] Hinterstoisser, Stefan & Pauly, Olivier & Heibel, Hauke & Marek, Martina & Bokeloh, Martin. (2019). An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Instance Detection.