Transfer Learning for Object Detection using State-of-the-Art Deep Neural Networks

J. Talukdar[*], S. Gupta[†], P. S. Rajpura[§], R. S. Hegde[§]
[*]Dept. of Electronics & Communication Engineering, Nirma University, Ahmedabad
[†]Dept. of Computer Engineering & Information Science, BITS Pilani, Hyderabad Campus
[§]High Performance Computing Lab, Indian Institute of Technology, Gandhinagar
Email- {[*]jonti.talukdar.05, [†]sanchit199617}@gmail.com

Abstract— **Transfer learning through the use of synthetic images and pretrained convolutional neural networks offers a promising approach to improve the object detection performance of deep neural networks. In this paper, we explore different strategies to generate synthetic datasets and subsequently improve them to achieve better object detection accuracy (mAP) when trained with state-of-the-art deep neural networks, focusing on detection of packed food products in a refrigerator. We develop novel techniques like dynamic stacking, pseudo random placement, variable object pose, distractor noise etc. which not only aid in diversifying the synthetic data but also help in improving the overall object detection mAP by more than 40%. The synthetic images, generated using Blender-Python API, are clustered in a variety of configurations to cater to the diversity of real scenes. These datasets are then utilized to train TensorFlow implementations of state-of-the-art deep neural networks like Faster-RCNN, R-FCN, and SSD and their performance is tested on real scenes. The object detection performance of various deep CNN architectures is also studied, with Faster-RCNN proving to be the most suitable choice, achieving the highest mAP of 70.67.**

Keywords—Transfer learning; Computer Vision; Deep Neural Networks; Synthetic Datasets; Artificial Intelligence

I. INTRODUCTION

Increasing automation has expanded the horizons of modern day computer vision systems, which in their present form, are not only limited to niche fields like robotics and manufacturing but also relevant in consumer areas like home automation, smart sensing, medical imaging, food industry, autonomous driving etc. [1]. This paradigm shift in the field of computer vision can be attributed to a few but revolutionary changes in computing trends in the past two decades. The first being advances in multi-core architecture as well as a subsequent increase in the clock speed of several state-of-the-art processor designs. Moreover, parallelization of data processing through advanced and dedicated GPUs has also played a critical role in increasing the bandwidth of data crunching process [2]. The second being the radical use of deep Convolutional Neural Network (CNN) architectures for the majority of tasks related to object detection and classification [3]. Earlier methods developed for object detection and recognition relied heavily on local and global feature descriptors like Scale Invariant Feature Transform (SIFT) [4], Histogram Oriented Gradient (HOG) [5], Space-time Interest points (STIP) [6] etc. and used linear classification methods like SVM and PCA in an end to end fashion. These feature engineering methods were highly application specific and difficult to customize, thus leading to sub-optimal performances over diverse datasets [7]. However, the use of neural networks revolutionized and highly automated the process of feature engineering.

One key reason for the popularity of neural networks is their generalization capability over a large amount of data. Thus, neural networks have the ability to approximate any random nonlinear function with minimal error [8]. Their independence over handcrafted features, as well as their high generalization potential, has led to their large-scale adoption in recent times. More than 60% of entries in the ILSVRC ImageNet object classification challenge [9] used deep CNNs as their base architecture [10]. The performance of CNNs relies on specific layer-wise characteristics, where multiple feature maps per layer are used to detect unique sets of spatially invariant features through convolutions. This allows CNNs to capitalize on excellent learning representations for the task of object detection and localization. Performance of deep CNNs is thus highly dependent on the various parameters and layerwise properties of the overall network, and with enough iterations and data, can be optimized to a very high level of accuracy.

CNNs rely on a large amount of data to attain their optimum level of accuracy. The development of high bandwidth object detectors catering to a large and diverse set of object classes has been possible majorly due to the inception of highly popular state-of-the-art datasets in recent times. These include the ILSVRC ImageNet dataset with more than 200 classes [9], the PASCAL VOC 2007 dataset [11] with over 20 classes and the COCO dataset [12] with close to 80 classes. Hence a dataset with a large number of diverse images is important for the complete and full-scale utilization of the CNNs in consideration. However, the annotation, collection and retrieval of such a dataset is a key impediment in the overall train-test object detection pipeline. In addition to being tedious, cumbersome and sluggish, there might be cases where the availability of data might be scarce. Thus, the critical task of dataset generation is extremely important to improve the quality of models being trained using deep CNNs.

We use synthetically rendered images to improve the overall quality of training data as well mitigating the challenges posed above. Transferring knowledge learnt from one domain to another is known as transfer learning [13]. We utilize pretrained models in conjunction with synthetic datasets to ensure effective transfer of features. Several instances of the use of synthetic data during the training process have surfaced in recent times. Peng et al. [14] have analyzed the effect of using images rendered from open source 3D CAD models with missing low-level cues like texture, pose, context etc. for training deep CNNs. They found that the ability of the network to learn effectively is unhindered despite using synthetic data with low 'cue invariance' and low degree of photorealism. Yosinski et al. [15] have analyzed the transferability of features through the different layers of CNNs, indicating that features during the initial layers show some degree of *generality* and can be utilized for multiple objects. This provides some leeway with respect to the specific set of features and characteristics of the synthetic data required for optimum learning. Additionally, Georgakis et al. [25] highlighted a joint approach of data augmentation where synthetic objects are superimposed on real scenes to improve dataset diversity. All the above methods used synthetic data as a form of secondary resource for enhancing the accuracy of the deep neural network.

In this paper, we analyze the performance of various state of the art deep CNNs [26] trained using 100% artificial datasets synthesized in an end to end pipeline and continue the work by Rajpura et al. [27-28]. These trained models are tested on never-seen real world images, specifically for the task of single class object detection. We also evaluate the effect of different synthesis strategies for datasets to identify specific characteristics which make one dataset successful compared to others, thus leading to *effective transfer* of learning representations or features from synthesis space to real space and improving overall performance. We finally aim to identify a holistic methodology where the overall object detection pipeline is optimized for the process of transfer learning through both the size and diversity of the synthetic dataset as well as through fine-tuning the hyperparameters of the deep neural network.

The rest of the paper is organized as follows: Section II describes the dataset synthesis strategies, Section III discusses the deep CNN architectures used as well as the overall object detection pipeline, Section IV talks about the results and Section V concludes the paper.

II. Synthetic Dataset Generation

Synthetic images form the key component of any transfer learning process. Networks trained completely on synthetic datasets show great potential and can be completely automated from start to end. To ensure greater scalability of this idea, we test the CNN's object detection potential on food items packed in a refrigerator environment. This scenario allows for great variety of data which can be placed in the refrigerator and proves to be an applicable use case where CNNs can generalize.

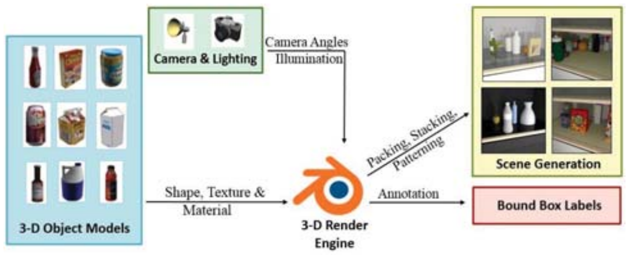


Fig. 1. Process flow description for Synthetic dataset generation through Blender Python API.

A. Process Methodology

All images are generated synthetically using Blender Python-API, an open source 3D rendering software. All images are RGB with an aspect ratio of 512X512 pixels. An object repository of close to 120 3D object models, custom built from the ShapeNet repository [16] is utilized for the process of scene generation. This includes objects like bottles, packaged food products, utensils, vessels, fruits, vegetables etc. all of which are categorised under the umbrella class of 'container'. A corresponding label file is generated in KITTI format for every rendered image and contains the bounding box co-ordinates for all true objects within the image.

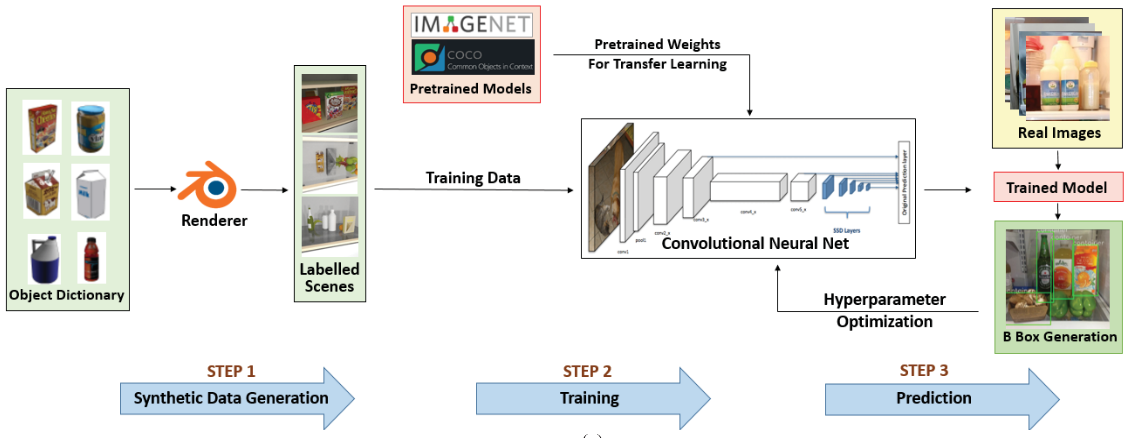
A number of parameters decide the diversity of the generated dataset as well as the impact it would have on the final object detection accuracy. This is discussed in detail in the incoming section. However, the key parameters which are varied to increase scene diversity include: medley of 3D models, different camera angles, shadows, variable lighting, texture & colour, object orientation and packing pattern etc. The overall process flow for synthetic dataset generation through 3D models and their annotation is shown in Fig. 1.

B. Key Data Generation Strategies

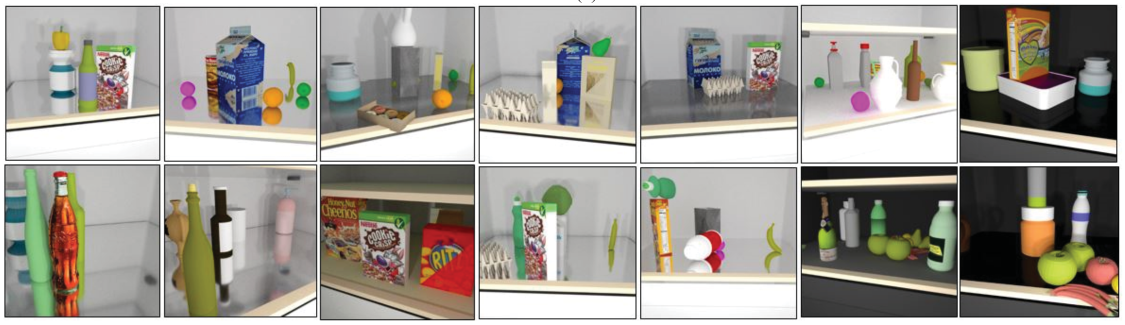
The ultimate aim of synthetic data generation is reduction of human time and effort. Hence we pursue the overall strategy of automating the data generation process. Images captured from real scenes encode a lot of information which is both unnecessary and difficult to recreate. Moreover, as discussed earlier [14], the synthetic data should target key transferrable features which are propagated across the layers of the CNN to improve overall accuracy. Thus, focus on photorealism is not an important factor in the generation of synthetic datasets.

Another important factor for optimal network performance is the presence of distractor objects in the training data. It is observed that data with distractor objects (non-containers) transferred the real scene information well compared to data without distractors. This equips the CNN with the ability to selectively discriminate true positives and reduce false detections, thus increasing the overall accuracy.

The design of our synthetic data generation algorithm also takes into account the overall patterning and clustering of objects in the scenes. Since real datasets contain some sort of



(a)



(b)

Fig. 2 (a). Standard Object Detection Pipeline for Transfer Learning. Step 1 involves generation of Synthetic Dataset for transfer learning. Step 2 involves the use of a state of the art CNN for training on the synthetically generated dataset. Step 3 involves testing on a real dataset to evaluate network performance and accuracy. (b) Different sample scenes from the generated synthetic dataset, which includes randomized packing, vertical stacking with variable camera angles and lighting.

variance in terms of randomness of scene density, it is essential to model this in the synthetic data to increase robustness of the data and prevent overfitting. We use pseudo-random packing of objects placed at randomized locations with different lighting, camera angles and positions. This increases scene diversity as well as provides natural clutter, thus modelling real data better.

Another key strategy we pursue is object scaling and x-y-z rotation. It is essential to estimate the object pose in synthetic data as closely as possible. This provides more positional variance to the synthetic data as well as increases the detector inference to variable orientations of the same object, thereby increasing precision. Finally, we also incorporate vertical stacking to model multilevel data in real scenes better. These strategies aim at holistic representation of data and can be scaled to any dataset and not just focused on refrigerator scenes.

Generating one training image using Cycles Render Engine takes an average of 50 seconds, which is highly dependent on the renderer and not on the quality of the image. This time can be further improved through multiprocessing and task scheduling practices.

C. Synthetic Datasets

In order to analyze the effect of the strategies stated above on CNN performance, we designed 3 types of datasets. In all of them the following things were randomized equally – camera angles, textures, object x/y positioning, shadows and lighting. The defining features of the 3 datasets are described below.

1. *Dataset I* – Containing 3576 images, each scene contained many objects, but with no distractor objects. All objects present were containers.
2. *Dataset II* – Containing 2750 images, each scene consisted of random number of containers, along with distractor objects such as fruits, non-containers etc.
3. *Dataset III* – Containing 3963 images, amalgamated the best of both datasets, including distractor objects, cluttered positioning, z-rotation (pose estimation) and vertically stacked objects

Fig. 2-(b) shows some sample images from the synthetic dataset. The ultimate goal is to find the optimal characteristics of any synthetic dataset in order to increase feature transferability as well as the object detection accuracy.

III. System Architecture

Fig. 2-(a) shows the overall object detection pipeline for transfer learning using synthetic data. The performance of the datasets is evaluated using four very recent state of the art deep CNN architectures [26] optimized for implementation in TensorFlow [17]. The CNN architectures used for training and testing are discussed below.

A. State of the Art Deep CNNs

A number of deep neural network architectures have been proposed in past [18], however, we have focused our implementation to the latest architectures which are fast, accurate and single-shot/single-pass detectors. We have tested the datasets on TensorFlow implementations of the following meta-architectures, namely- Faster-RCNN [19], SSD [20] and R-FCN [21], with a combination of various feature-extractors. We have used one or combination of Inception V2 [22], Resnet-101 [23] and Mobilenet [24] as the preferred convolutional feature extractors. These extractors are part of the deep CNN meta-architectures and responsible for *transfer* of features across the higher layers of the network. Their choice is highly application specific and determines the speed, accuracy and memory requirements due to major differences in their number of parameters and layer types. A review of the meta-architectures is presented below.

Faster-RCNN, an advanced version of Region based CNN performs detection in two stages. The first stage uses *Region Proposal Networks (RPNs)* to extract features from selected layers with the aim of hypothesizing bounding box locations. In the second stage, these bounding box proposals obtained in the previous stage are further cropped and processed downstream to evaluate localization of boxes by minimization of a loss function thus facilitating object detection. Thus, the RPNs help in narrowing down the search for relevant features related to the dataset.

SSD, (Single Shot Multibox Detector), eliminates the use of RPNs and detects objects by discretizing the output space with default boxes of various aspect ratios throughout the image and generating localization scores for each one. SSD uses multiple feature maps throughout the network in a concurrent fashion for making predictions. Since it uses a single deep neural network for detection, it is faster compared to other approaches with minimal accuracy tradeoffs.

R-FCN, (Region based Fully Convolutional Networks) works in a similar fashion to Faster-RCNN, but is computationally less expensive. Unlike Faster-RCNN, where crop features are computed in the same layers where object proposals are made, R-FCN takes crop features downstream to the last layer of the neural network. This reduces the overall sub computations for the same image.

All the three networks use an anchor box approach for prediction of bounding boxes [26]. Although different anchor generation methodologies like sliding window or regular grid are used, they all provide the advantage of being easily predicted using convolutional methods across the image due to shared weights in the network.

B. Training Methodology

All training and testing was done on an Intel-i7 5960X series processor aided with an NVidia GTX 1070. Since we focus on the test time performance of the networks, the training time is not a key factor in our study. In order to further reduce training time in transfer learning, we used open source detection models pretrained on the COCO and ImageNet datasets. Prior to training, pretrained models were used to initialize the weights of the CNN. We trained medium resolution models with 512x512 being the default train and test size. Data preprocessing included the conversion of generated KITTI labels and images to a single TF-Record format [18] compatible with TensorFlow framework. For Faster-RCNN model, the Resnet 101 feature extractor was used in 'atrous' mode. Thus the standard 16 pixel stride length was lowered to 8 pixels for enhanced results. The SSD detector was used with both Inception V2 as well as the latest Mobilenet feature extractors. The R-FCN meta-architecture was trained with the Inception-Resnet feature extractor for maximum accuracy. In most of the cases, the base learning rate was kept very low in the range of 0.004-0.007 and a multistep learning rate decay policy was used with a decay rate of 0.95. Additionally, the Smooth L1 loss function was utilized for localization loss.

IV. Model Evaluation And Results

In our case, all the trained models are tested on a validation set of 50 real crowd sourced images, thus allowing us to understand the transfer learning process. Bounding box visualizations are also analyzed in Fig. 4 for understanding the coverage characteristics of the deep neural networks. The transfer learning performance of the overall system is analyzed with respect to the performance characteristics of both synthetic dataset as well as that of individual networks below.

A. Evaluation Metrics

Object detection results are quantified using a popular metric known as **mAP** or mean Average Precision. It is evaluated as a product of recall (ratio of true positives to true positives plus false negatives) and precision (ratio of true positives to true positives plus false positives). Higher the mAP, better is the object detection performance of the network. State of the art neural networks in recent ImageNet challenges have successfully achieved mAPs of more than 75 [26].

The mAP is evaluated on the final set of bounding boxes at two IOU thresholds (Intersection over Union) of 0.5 and 0.7 respectively. IOU measures percentage or ratio of overlap of the predicted bounding box to the ground truth boxes. The upper bound of the IOU threshold (0.7) reflects the localization accuracy of the network while lower bound of the IOU threshold (0.5) reflects the detection accuracy.

The overall summary with the mAP, Precision and Recall values of all the networks with the corresponding datasets at IOUs of 0.5 and 0.7 are given in Table I and Table II respectively.

B. Dataset Performance

We can directly correlate mAP with the transfer learning performance as it decides the accuracy of object detection. From the tables I and II, it is clear that the *Dataset III* gives the best in terms of accuracy of detection, while *Dataset I* performs the worst. In fact there is a gradual increase in dataset performance

TABLE I. NETWORK vs DATASET PERFORMANCE METRICS @ 0.5 IOU

Meta-Architecture	Feature Extractor	Dataset I			Dataset II			Dataset III		
		mAP	Precision	Recall	mAP	Precision	Recall	mAP	Precision	Recall
R-FCN	Inception-Resnet	44.77	56.14	**79.75**	56.86	61.12	**93.03**	62.98	68.90	**91.40**
SSD	Mobilenet	0.49	11.76	4.24	50.65	73.76	68.66	39.10	84.26	46.40
SSD	Inception V2	10.24	51.00	20.07	43.05	87.16	49.39	27.02	72.66	37.18
Faster-RCNN	Resnet 101	**49.51**	**74.99**	66.02	**67.71**	**84.78**	79.87	**70.67**	**80.88**	87.38

TABLE II. NETWORK vs DATASET PERFORMANCE METRICS @ 0.7 IOU

Meta-Architecture	Feature Extractor	Dataset I			Dataset II			Dataset III		
		mAP	Precision	Recall	mAP	Precision	Recall	mAP	Precision	Recall
R-FCN	Inception-Resnet	33.82	48.31	**69.99**	40.02	51.01	**78.45**	51.04	61.99	**82.33**
SSD	Mobilenet	0.27	8.82	3.10	21.24	47.63	44.60	26.41	69.36	38.07
SSD	Inception V2	6.12	39.33	15.56	25.95	66.70	38.91	19.64	59.0	33.3
Faster-RCNN	Resnet 101	**41.80**	**68.72**	60.82	**53.98**	**75.15**	71.83	**62.54**	**76.08**	82.21

from *Datasets I-III.* It is noteworthy that the level of photorealism in all three datasets is the same. This is because objects imported for synthetic data generation in all the three datasets are common. However, there is an increase in performance of the *Datasets II* and *III* due to an increase in complexity of object clustering, placement and pose estimation. Visually, *Datasets III* is best in terms of closeness to real world data. The following key enhancements have increased the overall performance of *Datasets II* and *III*- Addition of distractor objects in synthetic scenes, stacking of objects in vertical plane, rotation and translation of objects in 3D space. Additional enhancements discussed in Section II-B have contributed to the best performance of *Dataset III,* increasing the average detection accuracy **by more than 40%** as compared to *Dataset I.*

We can also conclude that a high degree of photorealism is not at all essential for the process of transfer learning. In-fact the time and resources utilized in the generation of photorealistic images can be rather devoted to manual annotation. Instead, the key requirements that any synthetic dataset should fulfil include:

- Wide scale scene diversity to cover all possible scenarios during scene generation.
- Improve the number as well as quality of 3D models imported per scene.
- Dynamic rendering of scenes which cover all possible poses and variance compared to real scenes.
- An equally diverse amount of noise in the form of distractor objects to increase transfer rate of the overall scene.

C. *Deep Neural Network Performance*

From Tables I and II, it can be observed that the Faster-RCNN meta-architecture with Resnet 101 feature extractor provides maximum object detection accuracy, i.e. high mAP in all scenarios, while the different versions of SSD perform the poorest. Additionally, from Table I and II, it can be observed that R-FCN with Inception-Resnet framework provides the maximum value of recall while the Faster-RCNN version provides the good precision as well as recall.

This behavior can be explained in part due to the differences in the initial stages of all the three networks. While Faster-RCNN and R-FCN work on Region Proposal Networks, SSD works with default bounding boxes. Since RPNs themselves are fully convolutional in nature, the anchor boxes predicted for cropping and optimization in case of Faster-CNN and R-FCN are highly accurate in nature, while the default boxes predicted in SSD have no inherent advantage other than speed. This fundamental difference leads to comparatively higher performance of Faster-RCNN and R-FCN networks respectively. However, SSD gains an upper hand in terms of speed of execution as generation of default bounding boxes is computationally faster compared to RPNs. The performance variation also stems from the difference in the feature extractors used by the networks. In case of SSD meta-architecture, the use of Inception V2 and Mobilenet extractors speeds up the process of object detection. However, the Resnet-101 is the largest/deepest feature extractor thus leading to highest accuracy and time.

However, the flexibility of feature extractor selection with the above discussed meta-architectures provides great bandwidth for optimization when dealing with different case-specific custom datasets.

V. Conclusion

State of the art CNNs trained on artificially synthesized data promise great potential for transfer learning in cases with scarce availability of data. In such cases, the quality of synthesized dataset plays a key role in overall detection accuracy. Every dataset has its own key feature space based on its application domain. Although there is no marked way to define good dataset topology, there still exist methods like dynamic object scaling and translation, randomized positioning, variable lighting and camera angle, pose estimation, distractor noise etc. which increase the likelihood of a good accuracy score when

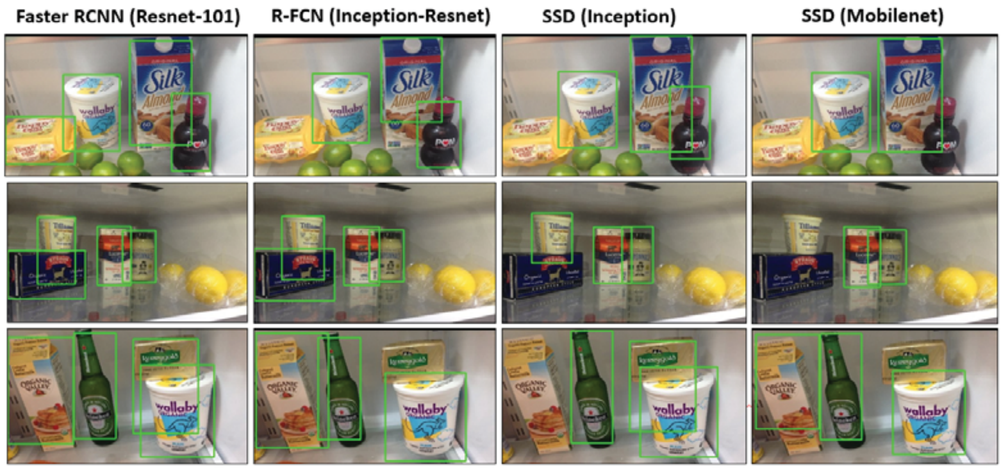


Fig. 3. Coverage output of each of the four networks on three different test images. Green boxes are the predicted box labels for networks trained on *Dataset III*. Faster RCNN (Resnet-101) predicts all the containers in the image with highest accuracy followed by R-FCN (Inception-Resnet).

validated on real world images. Hence it is essential to ensure that the synthetic dataset is well balanced in terms of its scene diversity, variance and noise. We evaluate the efficacy of these key strategies during the generation of synthetic datasets as well as analyze their transferability using state of the art CNN meta-architectures like Faster-RCNN, R-FCN and SSD and obtain very high detection accuracy. The overall transfer learning pipeline can further be improved by using a combination of real and synthetic data to achieve near perfection in application specific datasets. Additionally, the overall performance can also be improved by the use of an ensemble of different networks in an end to end fashion thus offering flexibility in terms of use of feature extractors and the corresponding meta-architectures. A combination of the above two strategies can be custom implemented for user specific datasets and multi class needs.

Acknowledgment

J. Talukdar and S. Gupta are thankful for the funding support from Indian Institute of Technology, Gandhinagar.

References

[1] M. Bojarski *et al.*, “End to End Learning for Self-Driving Cars,” *arXiv:1604.07316*, pp. 1–9, 2016.

[2] S. Chetlur and C. Woolley, “cuDNN: Efficient Primitives for Deep Learning,” *arXiv:1410.0759*, pp. 1–9, 2014.

[3] C. Szegedy, “Deep Neural Networks for Object Detection,” In *Proc. NIPS 2013*, pp. 1–9, 2013.

[4] Y. K. Y. Ke and R. Sukthankar, “PCA-SIFT: a more distinctive representation for local image descriptors,” In *Proc. IEEE. Comput. Vis. Pattern Recognition, CVPR,* 2004, vol. 2, pp. 2–9.

[5] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” In *Proc. IEEE. Comput. Vis. Pattern Recognition, CVPR*, 2005, vol. I, pp. 886–893.

[6] I. Laptev, “On space-time interest points,” in *International Journal of Computer Vision*, 2005, vol. 64, no. 2–3, pp. 107–123.

[7] K. Mikolajczyk, K. Mikolajczyk, C. Schmid, and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[8] S. Ferrari and R. F. Stengel, “Smooth function approximation using neural networks,” *IEEE Trans. Neural Networks*, vol. 16, no. 1, pp. 24–38, 2005.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, “ImageNet: A large-scale hierarchical image database.,” In *Proc. IEEE. Comput. Vis. Pattern Recognition, CVPR*, 2009, pp. 248–255.

[10] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556*, pp. 1–10, 2014.

[11] M. Everingham, L. Gool *et al.,* “The pascal visual object classes (VOC) challenge,” *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[12] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *Lecture Notes in Computer Science*, 2014, vol. 8693, no. 5, pp. 740–755.

[13] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Tran.on Knowledge and Data Engineering*, vol. 22, no. 10. pp. 1345–1359, 2010.

[14] X. Peng, B. Sun, K. Ali, and K. Saenko, “Learning Deep Object Detectors from 3D Models,” *arXiv:1412.7122,* pp. 1-3, 2014.

[15] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” In *Proc. NIPS*, 2014, vol 27, pp 1–9.

[16] A. X. Chang *et al.*, “ShapeNet: An Information-Rich 3D Model Repository,” *arXiv:1512.03012*, pp. 1–2, 2015.

[17] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” *arXiv:1603.04467*, pp. 1-19, 2016.

[18] A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” *arXiv:1605.07678*, 2016.

[19] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[20] W. Liu *et al.*, “SSD: Single shot multibox detector,” In *Lecture Notes in Computer Science*, 2016, vol. 9905, pp. 21–37.

[21] J. Dai, Y. Li, K. He *et al.* “R-FCN: Object Detection via Region-based Fully Convolutional Networks,” In *Proc. NIPS*, 2016, pp. 379–387.

[22] C. Szegedy *et al.*, “Going deeper with convolutions,” In *Proc. IEEE. Comput. Vis. Pattern Recognition, CVPR*, 2015, vol. I, pp. 1–9.

[23] S. Targ, D. Almeida, and K. Lyman, “ResNet In ResNet: Generalizing Residual Architectures,” In *Proc. ICLR*, no. 1, pp. 1–4, 2016.

[24] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Appl.,” *arXiv:1704.04861*, pp. 1-9, 2017.

[25] G. Georgakis, M. Arsalan *et al.* "Synthesizing training data for object detection in indoor scenes." *arXiv:1702.07836*, pp. 1-9, 2017.

[26] J. Huang, V. Rathod, C. Sun *et al.* “Speed/accuracy trade-offs for modern convolutional object detectors,” *arXiv: 1611.10012,* pp.1-21, 2016.

[27] P. Rajpura, R. S. Hegde, H. Bojinov, “Object detection using deep CNNs trained on synthetic images,” *arXiv: 1706.06782,* pp. 1-8, 2017.

[28] P. Rajpura, A. Aggarwal, M. Goyal, S. Gupta, J. Talukdar, R. S. Hegde, H. Bojinov, “Transfer Learning by Finetuning Pretrained CNNs Entirely with Synthetic Images,” In *Proc. 6th Nat. Conf. on Comp. Vis., Pattern Recog., Image Proc. And Graphics, 2017.*