Evaluating Mask R-CNN Performance for Indoor Scene Understanding

Badruswamy, Shiva
`shivalgo@stanford.edu`

June 12, 2018

1 Motivation and Problem Statement

Indoor robotics and Augmented Reality are fast becoming the fundamental building blocks of future home living. However, fast R-CNN evaluations on indoor images are little to non-existent due to lack of access to data driven mainly by privacy issues and cost of acquisition. A variety of fast R-CNN networks including Mask R-CNN exist for outdoor scene understanding. My motivation is to understand whether a modified, state-of-the-art Mask R-CNN would perform well on 3D projected to 2.5D (RGB+Depth dimensions) indoor, high-definition, 1080X1080 dataset.

The last few years have seen rapid development of fast scene understanding algorithms. However, many algorithms do scene understanding tasks on 2D data sets. The extension of fast R-CNN algorithms to RGB-D or full 3D data sets is in its infancy. Sophisticated NNs such as PointNet exist which first construct a 3-D point cloud or mesh and the 3D objects and instances on the mesh surfaces are annotated. While this approach gives us the best possible world co-ordinates and 360^0 panoramic depth, it is highly compute intensive and expensive to train and evaluate.

Recently, Facebook AI Research (FAIR) released code for a Mask R-CNN backbone which is state-of-the-art for 2D scene understanding tasks including instance segmentation. Matterport, a 3D imaging camera manufacturer, has adapted that code to release a more extendable MR-CNN code constructed on Keras and Tensorflow frameworks. Our goal is to investigate whether Mask R-CNN or a modified Mask R-CNN backbone on 2.5D RGB-D indoor images would perform well. We will further investigate whether hyper parameters or the network architecture need to be tuned differently to elicit the best performance.

2 Methods

Guided by the equirectangular 3D to 2.5D semantic data conversion approach described by Armeni et al., (2017)(http://3Dsemantics.stanford.edu/), we are proposing an architecture that trains a modified Mask R-CNN in 3 stages to take advantage of early stopping in each stage to cut down training and validation time.

In stage 1, we train the modified Mask R-CNN on just the backbone's Network heads. In stage 2, we fine-tune on ResNet layers 5 and up. In the final 3rd stage we further fine tune on all layers. This approach helps us stop training, if necessary, to just 1 or 2 stages, if a smaller network is able to remove bias and fit the 2.5D data well. In our experiments, we trained with $batchsize = 2$ (due to memory limitations) and the losses converged well enough for early stopping to kick in prior to 20 epochs in stage 1 and less than 5 epochs in stages 2 and 3.

The seminal part of the work required generating ground truth data (masks and bounding boxes) on images that were per pixel labeled at an instance level (color-coded instances). This effort involved heavy code development in two areas:

1. **Label Map generation:**For producing the label map, we relied on indexing every pixel's color value to a 256-base number. The color index of each pixel location is encoded as: $ColorIndex_{pixloc} = 256 * 256 * RGB_{pixloc}[0] + 256 * RGB_{pixloc}[1] + 1 * RGB_{pixloc}[2]$. Separately, a 1-Dimensional label array is created where each label is indexed as: $LabelArray[ColorIndex_{pixloc}] = LabelString$. The $LabelString$ is a concatenation of $LabelID, LabelName, Area, Location$ among other items. Each pixel's label is then obtained by returning the $LabelString$ attached to that pixel in the $LabelArray$. Our code for creating labeled masks can be found on our github.

2. **Binary Masks, RLE Byte Encoding and Bounding Boxes:**Binary masks are generated from semantic pixel maps by thresholding the pixel value to boolean 1 or 0 and an instance number is added to each mask instance. In our dataset, each image has only 1 object class but several instances of that object class. Since the Matterport camera captures several views of the same location (to build a panorama and the 3D point cloud), the same object class is captured in several images. This helps provide an augmentation effect to the images and enlarges our dataset. So, separate image augmentations were not used to further augment the images. After creating the binary masks, the instance masks in an image are stacked together and Run Length Encoded (RLE) as a byte string. Finally, ground truth bounding boxes are generated from masks using the rectangular co-ordinates surrounding the masks. The areas of the rectangular bounding boxes are considered areas of the masks for computing IoUs and overlaps.

3 Architecture search and Challenges

Our goal was to interpret indoor images when viewed by a mobile computer vision scanner such a robot or an AR device. We, therefore, needed a model that is computationally efficient to execute and also fast in its state-of-the-art frames per second detection capabilities. Our search led us to select Mask R-CNN , a step-up model from the faster R-CNN as the suitable model to investigate.

We also selected the data architecture afforded by the Stanford 2D-3D dataset due to its high-definition, high augmentation rate, less occlusion, indoor objects, and computation efficiency. We explored other datasets such as NYU-Depth V2 and Matterport 3D but we found these datasets to not have the seminal characteristics we were looking for as illustrated above.

The main challenge in this project came from the need to convert the Stanford 2D-3D dataset to a masked dataset similar to the ground truth MS-COCO instance masks and bounding boxes on which the Mask R-CNN (MR-CNN) was originally trained on. Further,upon conversion, sparse mask images needed to be weeded out. Since semantic pixels are not readily connected to annotations, class IDs, labels and such, several JSON files had to be generated to index and link the pixels to their labels and classes. Generating such link files and ground truth annotations took considerable amounts of time and code development. Refer to our github for all code files, outputs, configuration files, visualization codes and a host of other code and data we generated for this investigation.

The other main challenge was that the dataset has only 12 fore ground classes compared to over 90 classes for MS-COCO 2017 dataset. Therefore, the Resnet 101 based backbone Mask R-CNN over fit the dataset. To avoid over fitting, several modifications were made to the MR-CNN. The key modifications revolved around reducing the size of the network, adjusting learning rate to much lower than the 0.02 elected in the original paper, and gradient clipping and L2 regularization (weights decay) to prevent gradient explosion. Details of the modifications are listed below.

1. Smaller ResNet backbone: we found that incorporating a ResNet 50 gave us fast loss convergence and prevented bias and variance issues at the same time.

2. We applied a learning rate plateau smoothing as an adaptive learning rate to adjust to the directions of the loss momentum.

3. We exchanged the ADAM optimizer for a stochastic gradient descent with momentum. The baseline MR-CNN trained with batch sizes of 32, which we found to be computationally expensive. Therefore we decided to train with a batch size of 2, which meant that we did not have to apply batch normalization and could use stochastic gradient descent as our algorithm.

4. We also reduced the sparse entropy softmax activation nodes down to 13 from the 81 needed for the MS-COCO dataset.

5. Finally, we generated the more accurate synthetic bounding boxes from masks rather than the ground truth bounding boxes.

4 Dataset and Preparation

We used the Stanford 2D-3D Dataset. The dataset has 70,496 RGB images with corresponding segmentation images coded in Green and Blue channels. The images are 1080X1080, which were padded to 1088X1088 so they are divisible by 32 for ensuring the tensor shapes remained consistent. The dataset has 6 Areas, 13 object classes, 11 scene categories, and 270 scene layouts. From a shuffled population of 10,000 images from Areas 3 and 5a, we randomly chose 5000 training images, 1000 Dev, and 100 test images. All images, therefore, are from the same distribution and therefore we did not need to set aside examples for train-dev. Data is fed via a Keras data generator with batches shuffled for every step in the epoch.

5 Experiments and Evaluation

We took the original Facebook AI Research's MR-CNN with the full ResNet-101 backbone on MS COCO dataset and further fine tuned the backbone to evaluate on the Stanford 2D-3D dataset. Against this baseline, we measured our modified MR-CNN's performance on the Stanford 2D-3D dataset. We evaluate our results by research community-standard performance indicators such as mAP (mean average precision), mAR (mean average recall), and F1 score, per the formula below. mAP, mAR, F1 were computed for various IoU thresholds and then averaged.

$$F1 - score = (2 * mAP * mAR)/(mAP + mAR)$$

We will also evaluate comparison of the 3 MR-CNN losses (Class, Box, Mask) against the MR-CNN baseline tuned on MS-COCO dataset. The tables below summarize loss performance comparison between baseline and our modified MR-CNN versions.

Table 1: Evaluation Metrics Comparison

Model	dataset	mAP	mAR	F1 Score
MR-CNN Baseline	COCO	0.03	0.53	0.06
Modified MR-CNN Train	S2D	0.31	0.73	0.43

Table 2: Loss Performance Comparison

Loss Comparison	dataset	MR-CNN Box	MR-CNN Class	MR-CNN Mask
MR-CNN Baseline Train	COCO	0.5817	0.5912	0.5680
MR-CNN Baseline Val	COCO	0.5869	1.039	0.5583
Modified MR-CNN Train	S2D	0.7574	0.093	0.6964
Modified MR-CNN Val	S2D	0.7445	0.1103	0.6925

Our modified MR-CNN delivered a MR-CNN class loss of 0.093 for training and 0.1103 for validation against the baseline of 0.5912 and 1.039 respectively: a promising 1/6th reduction in training and 1/10th reduction in validation loss. Our losses are lower as our model involves training and evaluation on a low occlusion, simpler feature maps, 1 object per image dataset. Both training and validation losses converge in several hyper parameter tuning runs, indicating the backbone MR-CNN model's ready application to transfer learning for indoor scene understanding tasks. Our results were also superior for mAP measure: we obtain a mAP of 0.31 for our modified version versus 0.03 for the baseline. Overall, there is a 10X performance improvement, which demonstrates that MR-CNN can be readily applied to less occluded, lower classes indoor images as well, a promising result, which can be investigated further.

Further, our experiment runs showed that progressive learning rate decay coupled with gradient clipping (clipped at a gradient threshold of +/- 5.0) and reduction in network size were the key hyper parameters when tuned produced increased performance. Following these hyper parameters, we found that tuning IoU and non-max suppression thresholds to be very effective. We believe that further reduction in losses and increase in accuracies can be achieved by adding another input information channel viz. depth encoding, which could be the z-axis distance or surface normal vectors.

In summary, a smaller, modified MR-CNN performs well on Class loss and comparably on Mask and Box loss against the baseline. This ability to use a smaller MR-CNN is computationally very efficient compared to the expensive 3D point cloud based network.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girschick, Mask R-CNN, 2017, arXiv:1703.06870v3 [cs.CV]

[2] Iro Armeni, Alexander Sax, Amir R. Zamir, Silvio Savarese, Stanford University, UC Berkeley, Joint 2D-3D-Semantic Data for Indoor Scene Understanding.

[3] Thomas M. Breuel, DFKI and U.Kaiserslautern, Efficient Binary and Run Length Morphology and its Application to Document Image Processing, 2 Dec 2007, arXiv:0712.0121v1[cs.GR]

[4] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar,C. Lawrence Zitnick, Microsoft COCO: Common Objects in Context, 2014, arXiv:1405.0312 [cs.CV]

[5] chollet2015keras, Keras, Chollet, Francois and others, 2015, https://keras.io

[6] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Working code forked from Mask R-CNN for Object Detection and Segmentation, Waleed Abdulla, Github(matterport/MASK-RCNN)