CATER-based Object Detection and 3D Coordinations Estimation

Zhuo Zeng, Xuejin Tong, Chenxi Li

Abstract

This document provides a CNN-based Network to predict the object attribute. It starts from generating dataset, including developing a semi-automatic annotation tool and processing data in CVAT. With this dataset, we implemented an extended Mask-R-CNN model for attributes(e.g. color, shape, size, material) and 3D-Coordinates prediction.

1. Introduction

Neuro-Symbolic AI is increasingly relevant to tasks at the intersection of Computer Vision (CV) and Natural Language Processing (NLP). However, for some tasks, such as video-based dialogue, these datasets are still few and far between. In this project, we intend to use the recently released video-based diagnostic DVD dataset CATER (Girdhar et al., 2020) to process the scene parser for our upcoming neural-symbolic video dialogue model

https://rohitgirdhar.github.io/CATER/

Our project can be basically concluded as 3 steps:

- Generating a small dataset of annotated objects based on CATER videos, including developing a semi-automatic annotation tool and processing data in the Computer Vision Annotation Tool.
- Training an extended Mask R-CNN with the generated dataset for objection detection and recognition, which leads to getting the object attributes: color, shape, size, material, and 3D-Coordination.
- For 3D-Coordination prediction, implementing and training a CNN-based network for comparison.

After experiments, it can be proved that the extended Mask R-CNN we trained has a good performance.Code and data are available at:

https://github.com/levinz97/CATER

2. Related Work

Mask R-CNN: Mask R-CNN (He et al., 2018) is a general framework for object instance segmentation. The method extends Faster R-CNN (Girshick, 2015) by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. The model can efficiently detect objects in an image while simultaneously generating a high-quality segmentation mask for each instance. Mask R-CNN is simple to train and easy to generalize to some other tasks.

3. Generate Dataset

We collected data from All-Actions-Camera-Motion:

CATER new 005200 - 005499 .avi

with a fixed camera position.

In each video, capturing the 7 frames in time order and analyzed the frame image. In each image, there are several objects with different shapes, colors, sizes, materials and 3D-Coordinates. For each object, collecting its Segmentation and Bounding box position in order to locate it in the picture. All data were finally saved as Coco 1.0 Format.

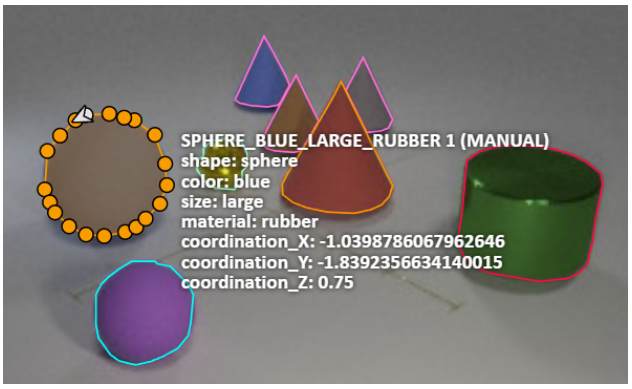


Figure 1. Labeled Image

3.1. Semi-Automatic Annotation Tool

Apparently labelling 2000 images manually is a boring work. To avoid wasting time we developed a semi-automatic annotation tool, which is able to detect and recognize objects in the image. This tool is based on conventional Computer Vision and Machine Learning.

3.1.1. OpenCV: GrabCut

Failure Experiment: In conventional OpenCV there are plenty of methods that can be used to detect the objects automatically. At first we tried the method "K-means" and "anisotropic diffusion+thresholding in HSV space ". But the results were not optimal.

Successful GrabCut: In the end, we chose an interactive foreground extraction method: Grabcut. With this method we were allowed to manually subtilize the detail after automatic segmentation A Gaussian Mixture Model(GMM) is used to model the foreground and background. Combining GMM segmentation and manual input Bbox the algorithmus would iteratively approach the accurate segmentation

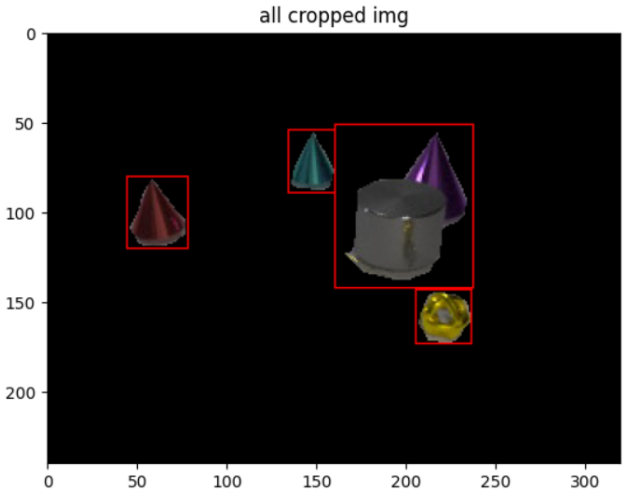


Figure 2. Result from initial Grabcut. Two close dark objects were detected as a big one

Manual Selection: Our algorithm would initially generate some segmentation. Most of them were fine but some details were not acceptable. Then we can manually select region to separate the object boundary from background and other objects.

3.1.2. Simple attributes classifier

Color, Size, Material: Through Grabcut we can get the relevant parameters of each object, such as HSV value, area, contour length, contour center and Bbox.

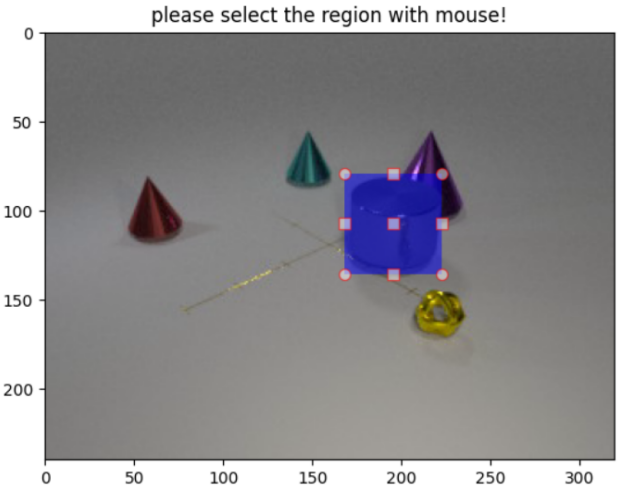


Figure 3. Manually selecting target region

Based on these we designed a simple classifier. After counting the parameter information from 50 images and used some simple classification models (e.g. Support Vector Machine(SVM), RandomForest, and Multilayer Perceptron(MLP)) in the scikit-learn library. To improve the accuracy of model predictions, we apply ensemble learning to predict color, material, and size. As a result, The accuracy of the predictions is about 90%

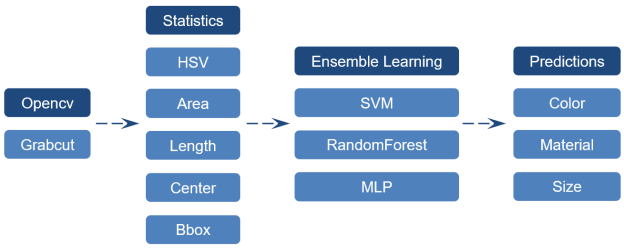


Figure 4. Simple Classifier for attributes color, size and material, no shape

Shape: However, simple machine learning models for shape prediction cannot achieve good results, so we have to manually input shapes.

3.1.3. Implement Mask RCNN

Through the previous method, we have completed the annotation of about 1000 images. Using these images we trained a Mask Rcnn Model and applied it to pre-segmentation and attribute prediction. The Model has higher pre-segmentation accuracy, which can separate objects from the background and shadows better. At

the same time, it solves the problem that Grabcut cannot effectively separate connected objects.

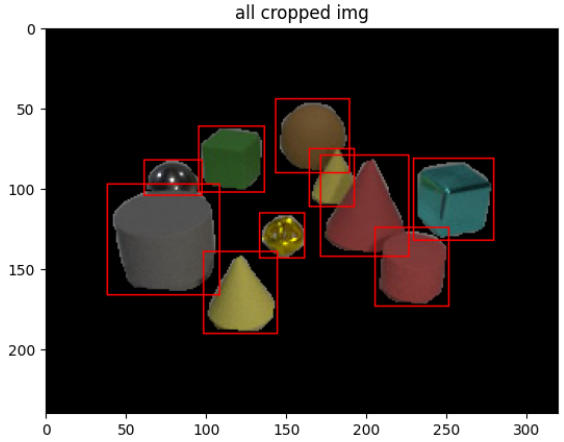


Figure 5. Mask RCNN improves the accuracy of detecting segmentation

3.2. Check Attributes and Assign Coordination

Finally we can cross check our raw predicted attribute with original data from CATER video. If we found the corresponding object, we would assign the coordination to our predicted one. Otherwise, the coordination would be [0,0,0].

Twins Object: It occurs that two identical object exist in one image and twins would be given same coordination, which would influnce our 3D-Coordination prediction later. For these "Twins Object" we programmed to locate them in the exact frame then manually input their correct coordination.In the other way, it suggests that our final CNN based Predict Model is able to detect coordination of identical objects.

Wrong property: In the following steps we would concentrate on the [0,0,0] object and change the attribute in CVAT

3.3. CVAT: Computer Vision Annotation Tool

CVAT is free, online, interactive video and image annotation tool for computer vision: https://cvat.org.

3.3.1. Transfer the predicted data into COCO Format

We can upload our raw predicted data into CVAT as long as the data format can be accepted by CVAT. So we designed COCO Transfer which allows us transfer our data as COCO 1.0 Format

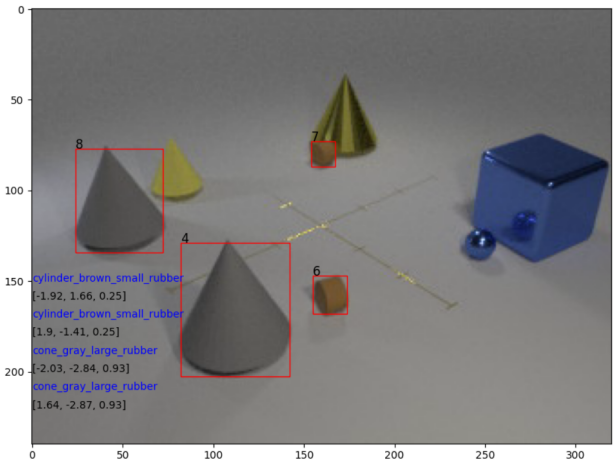


Figure 6. There are two pairs of twins object

3.3.2. Refinement in CVAT

The pre-segmentation of images have lots of detailed problems: segmentation involves too much background: Similar and closed objects were detected as a big single one: segmentation boundary is not smooth enough; wrong predicted properties... We can solve all of these problems in CVAT

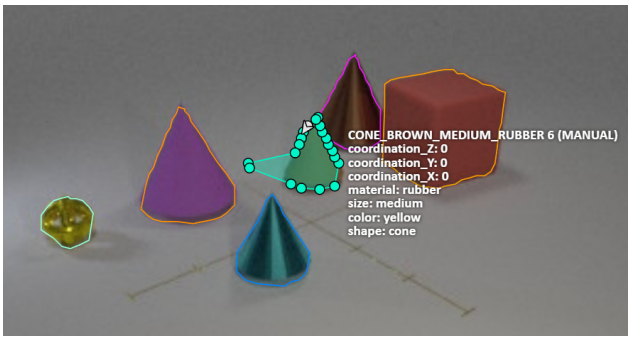


Figure 7. In CVAT the inappropriate segmentation and wrong property would be fixed. The object with wrong property has coordination [0,0,0]

3.3.3. Dataset

Label: Labels include 5 shapes, 9 colors, 3 sizes and 2 material. 193 kinds in total.

Dataset: The final dataset have 2044 images and 13648 objects in total

4. Compact Attributes and 3D-Coordinates Prediction

In this section, we introduce the details of our proposed approach.

- The results of standard Mask R-CNN Model for Attributes prediction.
- Introduction of the Baseline Method used for Coordinate prediction
- Introduction of our Extended Mask R-CNN Model for compact attributes and 3D-Coordinate Prediction.

4.1. Attributes Prediction

For this part we need to predict of attributes including the shape, material, color and we formulate this as an objection detection problem, i.e., we assign each combination of the attributes an exclusive class ID. And we use the standard Mask R-CNN to do the instance segmentation as well as bounding box prediction. For this part the standard Mask R-CNN has a very good performance with Bounding box prediction 80 AP and Instance segmentation reaches 85 AP on the test dataset, even though the training dataset has only about 1700 pictures. So, in the following parts we do not try to further improve the performance of this part but focus on the 3D coordinate prediction.

```
[01/25 23:15:53 d2.evaluation.testing]: copypaste: Task: bbox
[01/25 23:15:53 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[01/25 23:15:53 d2.evaluation.testing]: copypaste: 80.5750,99.4326,97.5292,78.9643,84.4631,nan
[01/25 23:15:53 d2.evaluation.testing]: copypaste: Task: segm
[01/25 23:15:53 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[01/25 23:15:53 d2.evaluation.testing]: copypaste: 85.7466,99.3078,98.5934,81.7285,92.0653,nan
```

Figure 8. The first 2 tasks in regular Mask RCNN

4.2. Baseline model for 3D-Coordinates Prediction

The estimation of the 3D coordinates of an object in a 2D image is an important topic in computer vision field. Due to lack of relevant information in the single image, approximating the 3D coordinates can be considered as a difficult task.

One of the common strategies is based on geometric computer vision. This strategy estimates the 3D coordinates of the object by using the projection matrix. Since we can not get accurate camera parameters from our dataset, this strategy is not applicable to our project.

On the other hand, the development of deep learning provides new methods for solving problems in computer vision field. This work introduces two solutions to predict the 3D coordinates of an object by using CNN model in deep learning. The first one is the Baseline model, which purely based on Pytorch, and the second one is an extended Mask R-CNN model.

The strategy of this model is to use the raw image and the corresponding box of objects in images to predict the 3D coordinates through a CNN network. We will explain the detail of the baseline model in the next subsection.

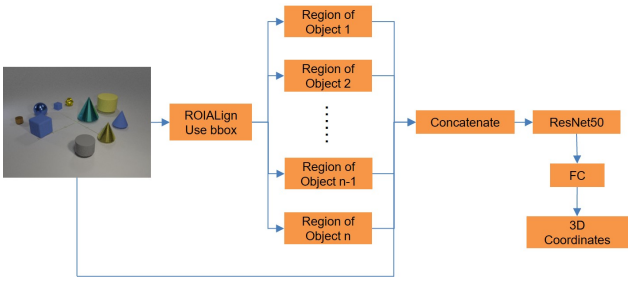


Figure 9. Baseline Model

4.2.1. Data Preprocessing

The dataset used for the 3D coordinates prediction task is from the dataset, which we have generated in the second part. From dataset we can get the raw image, corresponding bounding box and the ground truth (3D coordinates). According to the ROIAlign method in mask rcnn, we can extract the region of objects within the bounding box. There are two methods to concatenate the region of objects with raw image. In the first one we keep the shape of region and fill the vacant part with 0 to ensure the same shape as the picture. In the second one we simply resize the region of objects to the size of the image. Then we can get the input image with 6 input channels. Since the second method got the better performance, we chose that method for data preprocessing.

4.2.2. Network Design

To reach a performing solution, we used different CNN models to extract the image feature, including the classic CNN model resnet18, resnet34, resnet50 and resnext. After many attempts, by using resnet50 we almost got a better result. On the other hand, in Mask R-CNN model resnet50 is also used to extract the image feature. Therefore, we can compare the performance of two models later.

Resnet50: A classic CNN model for classification, but 3D coordinate prediction is a regression task. We need to use transfer learning to modify the model. Since

the input is a image with 6 channels, we modify the parameter of the first layer of the Resnet50 model. In the end. We changed the fully connected layer of the Resnet50 model. We employed two fully connected layers to output the 3D coordinates.

4.2.3. Hyperparameter

To train the model we also need to optimize hyperparameters. For batch size, small batch size led to large fluctuations in loss. But for large batch size, we need to consider the RAM constraint. For optimizer, in our task the training with Adam Optimizer cannot well converge. So we use other optimizer. And we also trained model with different learning rate. After many attempts, we adopted the following hyperparameters. Training with batch size 8, stochastic gradient descent(SGD) optimizer, and learning rate 0.001.

4.2.4. Loss

Since it is a regression task, we adopted the most commonly loss function mean squared error(MSE). For training, In the curve of training loss, there are some large fluctuations. Since the number of objects in different images varies greatly, it may indicate that each batch is not homogeneous. The data in some batches are quite different from the data in other batches. In addition, the training loss in the graph is less than 0.2 and in our test dataset the test loss is about 0.3.

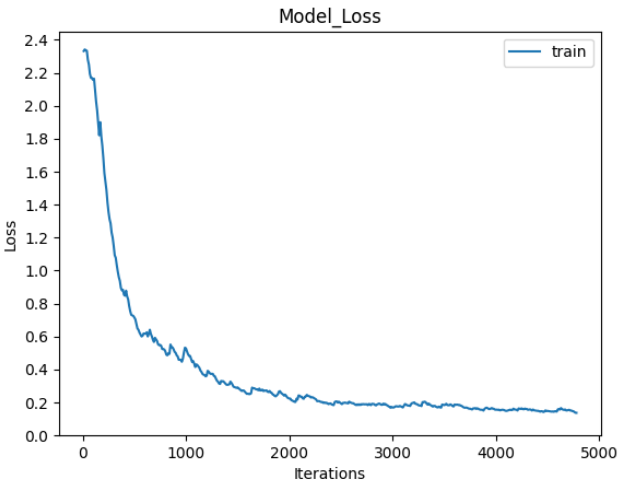


Figure 10. Train Loss for Baseline

4.3. Extended Mask R-CNN Model

Inspired by lots of existing models like DensePose[Neverova et al., 2018] and MeshRCNN[Hanocka et al., 2019] which exploit the potential of Mask R-CNN for other tasks, we proposed a novel model to achieve the 3 tasks:

- instance Segmentation
- Bounding Box prediction
- 3D-Coordination Prediction in one single model, instead of two for separate objection detection and coordinate prediction.

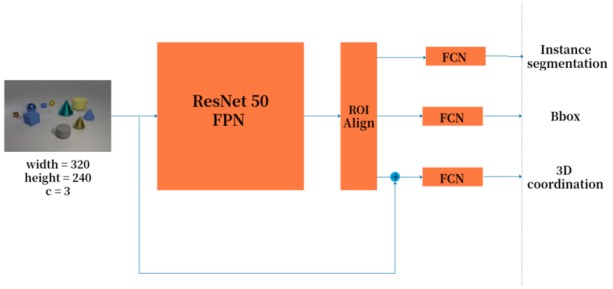


Figure 11. Achieving 3 tasks in one single model

4.3.1. Primitive Idea for 3D Coordinate Estimation Head

The similar idea from baseline model: use raw image concatenate with bounding box proposed by RPN without any backbone features in FPN. However, we cannot afford to have a very complex model like Baseline which uses a ResNet50 for this part, because we need to train Mask R-CNN and the V-RAM only has 8 Gigabytes, and for training the baseline model alone, its consumption is more than 7.7 GB. So, for the simplicity, we only apply a stack of convolution layers with kernel size 3 followed by batch normalization layer and Leaky ReLu, we have totally 5 such blocks. From the recorded training losses, we can see that, coordinate loss decreases dramatically in the first 500 iterations, however, the model fluctuates at loss = 2.0 for the rest 3500 iterations (much worse than baseline 0.3). The further architecture improvement of this part will be discussed in detail later.

Figure 12. Training Loss

Algorithm 1 Analytical Solution

Data: Layer parameters k_l, s_l, p_l for $l = 1, 2, ..., L$
Result: Calculate the receptive field size r
$r = 1$;
$S = 1$;
for $l \leftarrow 0$ to L do
  for $i \leftarrow 0$ to l do
    $S = S * s_i$;
    $r = r + (k_l - 1) * S$;
  end for
end for
Return r

4.3.2. Model Improvement: Lager Receptive Field

It is straightforward to use the analytical solution to calculate the receptive field of the input layer, and with larger stride and dilations, the receptive field increases much faster.

```
kernel_size 3 dilation 1, stride 1, n_layers 4
cumprod_stride=[1 1 1 1]
layer2 has receptive field 5
layer3 has receptive field 7
layer4 has receptive field 9
9
kernel_size 3 dilation 1, stride 2, n_layers 4
cumprod_stride=[ 2  4  8 16]
layer2 has receptive field 7
layer3 has receptive field 15
layer4 has receptive field 31
31
kernel_size 3 dilation 2, stride 2, n_layers 4
cumprod_stride=[ 2  4  8 16]
layer2 has receptive field 21
layer3 has receptive field 85
layer4 has receptive field 341
341
```

Figure 13. With larger stride and dilations, the receptive field increases much faster

But why increased receptive field helps? In the image below, it is clear to see that, to estimate the 3D coordinates of object, it does not make sense if the receptive field of model is extremely small.

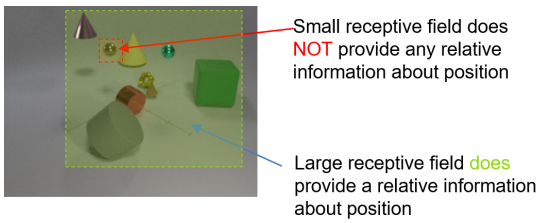


Figure 14. Comparison different sizes receptive field

For example in red rectangle, it does not provide the model with any positional information but only the local features about its shape color material etc., what we want is that the model can perceive at least some neighboring regions, which contains some global information about where it is, for example in green rectangle. And our assumptions are proven by the following comparison, and we would like to further explore the effects of receptive field in Experiment by using different levels of backbone features.

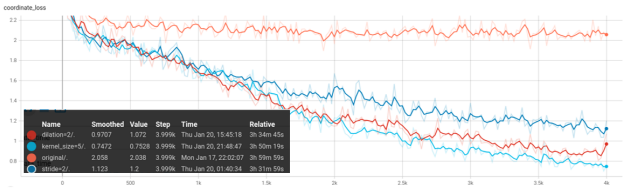


Figure 15. With increased receptive field the training loss improved from 2.0 to 0.7 but still worse than baseline (0.3)

4.3.3. Model Improvement: Use the Backbone Features from Mask R-CNN

The model in previous part is still not usable for 3D coordinate estimation. So, the further improvement is using the additional Backbone features shared for object detection and instance segmentation, but Backbone features have 256 channels, it is too large to feed directly to over coordinate prediction head.

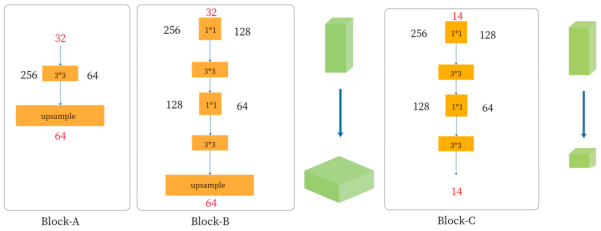


Figure 16. Block-A: directly apply 3x3 Conv -> training time 9 h; Block-B: First 1x1 Conv to decrease channels, followed by 3x3 Conv on lower dimensions features -> training time 5 h; Block-C: instead of upsample the backbone features, we apply an additional encoder for the image and bounding boxes.

The image and bounding boxes are reshaped to the size of 128 x 128, in order to be able to concatenate with the low-resolution backbone features of 32 x 32, we use a Decoder block to upsample to higher resolution and decrease the number of channels, so that it can be concatenated with raw image and bounding box from the

image. For this part, we also tried to keep the aspect ratio of bounding boxes and image and padded with 0s, but it does not have significant impact to the model performance. This let us to doubt the importance of image and bounding boxes, which will be further explored in Experiment. Using additional FPN features improves the training loss dramatically from 0.74 to 0.15, however, the training time also increases from 3.8 hours to more than 5 hours.

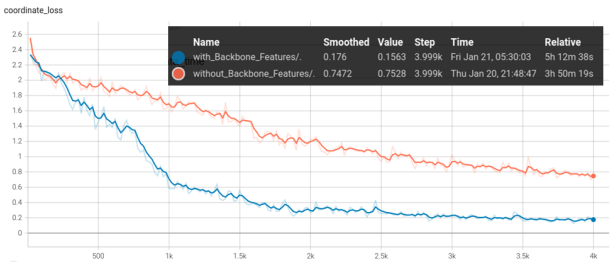


Figure 17. Use Backbone features from ResNet50 FPN

4.3.4. Model Improvement: Different Architectures for Coordinate Head.

The concatenation of backbone features and raw image with bounding boxes will be fed into the coordinate head. The primitive coordinate head has very poor performance and we learn that larger receptive field can improve the accuracy. So, we have designed following 4 different blocks for comparison. We want to emphasize the block 4, where we apply the 1x1 convolution firstly to decrease the number of channels and then apply the dilated convolution with different dilations in lower dimensional features, its influence will be shown and discussed in the following experiment part. And inspired by the ResNext[Zhou et al., 2021.], we can replace the last separate 1x1 convolutions then addition with early concatenation and a single large 1x1 convolution. This can further optimize the training time.

5. Experienment

In this section, we describe the experimental details and compare the performance of our model with Baseline. Moreover, we preform the ablation studies to explain the how the architecture of coordinate prediction network as well as selection of different backbone features influence the performance. Lastly, we explore what the model really learns from the image and features.

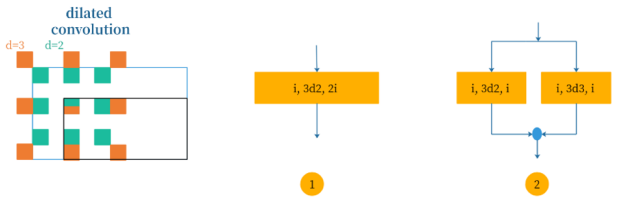


Figure 18. Apply Dilated Conv (e.g., kernel 3x3 with dilation 2 instead of kernel 5x5); Concatenate output from the kernel with different dilations

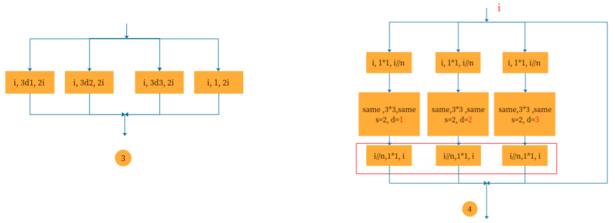


Figure 19. Increase the cardinality (number of kernels); Apply 1x1 Conv to decrease the channels and apply the dilated 3x3 convolution on lower dimensional features.

5.1. Dataset

For the jointly attribute prediction and 3D coordinate estimation we use the CATER dataset which are from a synthetic dataset and are labeled by ourselves for its attributes and 3D coordinates (contained in original json annotations from CATER). Its resolution is 240x320 @ 3 channels (RGB image). The training dataset contains about 1700 images, and test dataset contains about 340 images. These training datasets are from about 250 video sequences and test from another 50 video sequences, this ensures that the test dataset are totally new images which are never seen by our model during the training.

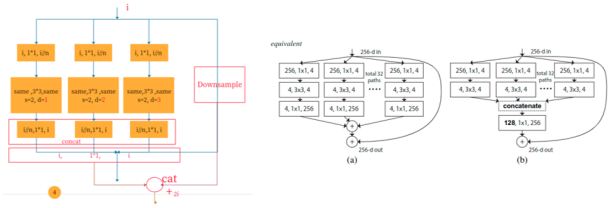


Figure 20. Left: improve the block 4 inspired by ResNext[reference]; Right: ResNext* block, (a) Aggregated residual transformations, the same as (b): A block equivalent to (a), implemented as early concatenation.

5.2. Training of Model

For the training of part, the model is implemented in PyTorch framework. The initial weight for standard Mask R-CNN is provided by Detectron2 which is mask_rcnn_R_50_FPN_3x. We train our model on NVIDIA GeForce GTX 1080 8G graphics card. Batch size is 7. Since we need to predict the size shape color and material jointly, we do not apply any data augmentations, otherwise the attributes will be changed without correspondence changes in ground truth (However, random flip is proven that does not affect the 3D coordinate prediction). We trained the model totally 4000 iterations, with linear warmup 500 iterations and we decreased the learning rate to 80% at 1500, 2500, 3500 iterations. The initial learning rate is 0.01 and optimizer is SGD (since we are training on a small dataset, SGD has better performance over Adam given a suitable learning rate). The training loss of 3D coordinates is mean squared error (MSE).

5.3. Comparison of different blocks

With comparison of different models, we can learn that the grouped dilated convolution and the trick of applying 1x1 convolution before larger kernels have great improvement to the model accuracy and training time reduction. The best result is about 64% better than the baseline method and only needs about 80% of training time. And we would like to point out that, this is still not the final performance of our model, the final result is shown in Conclusion.

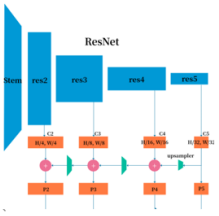
Block	MSE Loss	Training Time	Comments
1	0.23763	4:18:39	Simple k=3, d=2
2	0.22317	4:28:42	concat d=2, d=3
3	0.21262	4:37:42	concat d=1, d=2, d=3
4	**0.17624**	**4:09:04**	Grouped Conv 1x1 followed by 3x3 d=1,2,3
Baseline	0.30595	2:01:02	Only 3d coordinates prediction (additional 3 hours for mask rcnn)

Block	MSE Loss	Training Time	Comments
original	0.17624	4:09:04	Grouped Conv 1x1 followed by Conv 3x3 d=1,2,3
Variant 1	0.13322	4:03:19	Grouped Conv 1x1 followed by Conv 3x3 d=1,2,3,4
Variant 2	0.18590	3:55:16	replace last group dilated conv with dilated ResNext block
Variant 3	**0.10897**	**3:43:18**	Decrease number of Blocks from 4 to 3, d=1,2,3

Figure 21. Comparison of different blocks and baseline method; Further improve the Block 4 with hyperparameter tuning

5.4. Best Backbone Feature to use

In order to find out which backbone feature is best for the 3D coordinate prediction we performed a comparison on different level of Backbone features, this also represents different receptive field. The most noticeable result is that after applying the P4 features we can achieve 67% improvement over same scale C4 which does not contain any high-level features. This also proves that the feature pyramid network that contains the features from different scales can have a significant impact on the tasks that requires some global information, e.g., 3D coordinate prediction.



Features	MSE Loss	Pooler size
C4	0.29374	14x14
P4	0.09576	14x14
P2	0.10246	32x32
P5	0.09592	7x7
ALL	0.10027	32x32

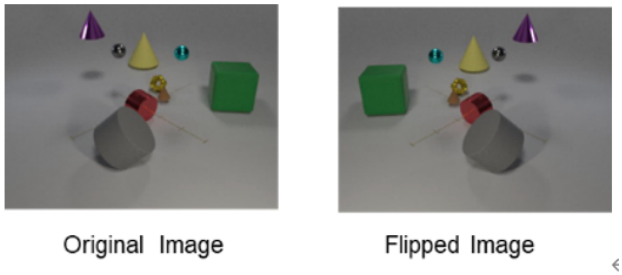
Features	MSE Loss	Pooler size
ALL	0.10027	32x32
ALL	0.10298	7x7
ALL	**0.09080**	14x14
ALL	0.09550	18x18
ALL	0.09764	24x24

Figure 22. Architecture of ResNet with FPN; We choose the best pooler size of 14x14 at last.

5.5. What has the model really learned?

In order to find out if model only learn to memorize different regions on image instead of learning the relations between objects, we apply additional random flip during the training, since the camera viewpoint is fixed, the flipped image has an opposite reflection, shadow orientation, etc., we would like to find out if the model really learns such information other than simply memorizing bounding box locations on the image. The result shows that, even though the training loss decreases slower if we enable the random flip, the final training as well as the test loss are very close to each other. This shows that the model learns to use the relations that are equal variant to the flipped image. However, to find out what the model really learns, further comparison and ablation study is needed, which can be seen as complement experiment to this project. Does the model only learn to produce the same value when bounding box in a certain location in an image (e.g., center right) instead of learning relative informa-

tion like orientations of shadows, occlusions, lines on the floor, etc.? In order to find out if model only learn to memorize different regions on image instead of learning the relations between objects, we apply additional random flip



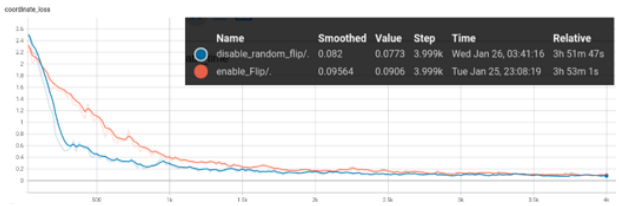


Figure 23. Loss decrease during training, with enabled random flip, model converges slower.

Random Flip in Training (p=0.5)	MSE Loss
Enabled	0.110928
Disabled	0.111304

Figure 24. Final MSE loss after random flip

6. Conclusion and Final Result of our Model

We can see that the major computational complexity and number of parameters come from the model of ResNet 50, and with our proposed model where the backbone features are shared among different tasks, we successfully avoid having a second ResNet model for the 3D coordinate estimation, which will otherwise increase the V-RAM consumption as well as training time dramatically.

In conclusion, we proposed a compact model to achieve attributes prediction as well as 3D coordinate estimation simultaneously, and this model not only decreases the training time and memory footprint, but also increase the 3D coordinate prediction accuracy over a separate model by selecting correct features and improving architecture design.

Features	Computational complexity	Number of parameters:
Coordinate Head	0.01 GMac	77.53 k
Encoder	0.06 GMac	14.02 k
Decoder	0.06 GMac	241.06 k
ResNet50	6.38 GMac	25.56 M

Comparison	Model Parameters	MSE Loss
Baseline	27.26 M	0.30595
Proposed method	**0.33M** + 25.56 M (shared backbones)	**0.08211**

Figure 25. Model complexity and number of parameters; Final comparasion of proposed models and Baseline Model

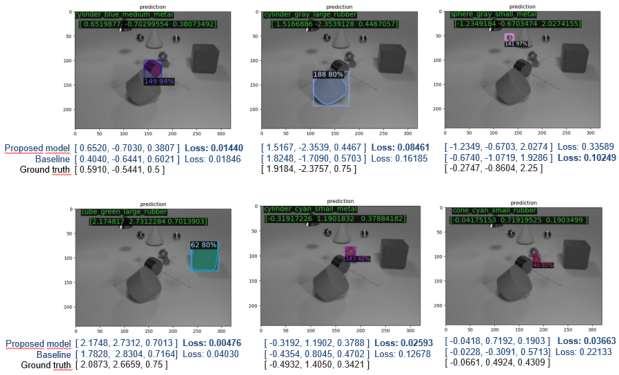


Figure 26. CNN based 3D Coordinates Prediction

References

Hung Le, Chinnadhurai Sankar, Seungwhan Moon, Ahmad Beirami, Alborz Geramifard,and Satwik Kottur. DVD: A diagnostic dataset for multi-step reasoning in video grounded dialogue, August 2021.

Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions and temporal reasoning, 2020.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

Natalia Neverova, Riza Alp Guler, Iasonas Kokkinos. Dense Pose Transfer, 2018.

Tianyan Zhou; Yong Zhao; Jian Wu. ResNeXt and Res2Net Structures for Speaker Verification, 2021.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, Daniel Cohen-Or. MeshCNN: a network with an edge, 2019