MASTER THESIS

GENERATING SYNTHETIC TRAINING IMAGES FOR INSTANCE SEGMENTATION USING SALIENT OBJECT DETECTION AND IMAGE COMPOSITIONS

Pratik A. Naik

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF BIOMECHANICAL ENGINEERING

EXAMINATION COMMITTEE
dr. E.H.F. van Asseldonk
dr.ir M. Vlutters
dr. N. Strisciuglio

DOCUMENT NUMBER
BW - 798

UNIVERSITY OF TWENTE.

JUNE 2021

ABSTRACT

Instance segmentation is the task in computer vision where each object in an image is localized, identified and given a pixel-level segmentation map. Instance segmentation is used in applications such as autonomous driving and robotic manipulation. Deep neural network models have performed well on the instance segmentation task. However, these deep neural networks require a lot of annotated data for training. This data is usually manually annotated. This manual annotation process is expensive and time-consuming. Manually annotating is difficult for an individual researcher or a small research group because there is no way to determine the number of images needed to sufficiently train a model. To solve this problem, a synthetic image generation pipeline is proposed and tested for the instance segmentation task with Mask R-CNN models.

In this study, synthetic images and their annotations are generated using foreground extraction and image compositing. A salient object detection network called U^2-Net is used for the foreground extraction step. Images are composed with the extracted foregrounds using operations like random flipping, scaling, and rotating. Along with this, the effects of adding noise, adding unlabelled instances were studied. The effects of using hybrid datasets and initializing training with synthetic data and then retraining the model with real image data were also studied.

Generating synthetic image datasets required 20% of the time needed to manually annotate images. However, models trained on synthetic images with added noise and unlabelled instances have, on average, 50% of the performance of the model trained on real image datasets. Models trained on hybrid datasets which contain both synthetic and real images do not have any benefit as these have performance almost equal to their real image subsets. Retrained models were initially trained with synthetic or hybrid datasets and then retrained with real images. Retrained models performed better than the model that was trained on only the real images.

ACKNOWLEDGEMENT

There are a lot of people that have helped and supported me in bringing this assignment to fruition. Special thanks to the following people.

I would like to extend my deepest gratitude to my parents, Anil Naik and Vaishali Naik, for all of their support and love. I am grateful for all the opportunities that they provided for me.

I am also very thankful to Dr.Ir. Mark Vlutters for the opportunity to work on this assignment. Completing the assignment would not have been possible if not for his insights and feedback. Thank you for allowing me to steer the assignment.

I would also like to thank Dr. Edwin van Asseldonk for providing the feedback to improve the report. Also, thank you to Dr. Nicola Strisciuglio for his valuable feedback on the assignment.

Finally, I would like to thank my housemates and my friends for their support and encouragement which kept me going.

CONTENTS

1 INTRODUCTION

1.1 Problem Background

Computer Vision research aims to create algorithms that can perform functions of the human visual system[1]. Automating the function of the human visual system allows for applications where computers can perform tasks such as object tracking [2], path planning in robotics [3], object or human pose estimation [4], and detecting cancer cells [5].

Image Classification is the task in computer vision research where a single object in the given image has to be recognised. The most popular example for image classification task is the MNIST dataset [6]. It is used for handwritten digit recognition. Other image classification datasets are CIFAR-10 [7] and Caltech-101 [8]. Each Image in the datasets for this task only contain one object per image.

Object Detection is the computer vision task where for the given image, objects have to be recognised and localized. Deep neural networks are usually employed for object detection task. These deep neural networks require a lot of training images. For the object detection task, the training images have to be annotated with a bounding box around the object and the object class. KITTI [9] and Pascal VOC[10] are datasets used for object detection. Object detection task is class-aware and instance-aware.

Semantic segmentation is the computer vision task where for a given image, object classes have to be recognised and a segmentation mask for all the pixels in the image has to be generated. The semantic segmentation task is class-aware but not instance-aware. This means that no information is provided about the number of occurrences of the given class in the image. ADE20K[11] and PASCAL context[12] are datasets made for semantic segmentation tasks.

Instance Segmentation is the computer vision task where each object instance in the image is segmented at the pixel level. It differs from Semantic segmentation since semantic segmentation is not instance-aware but it does provide pixel-level segmentation for the image. Object Detection or localization tasks are class-aware and instance-aware but do not provide pixel-level segmentation. Instance segmentation is a combination of these tasks since it is class-aware, instance-aware, and provides pixel-level segmentation.

Deep neural networks are used to perform instance segmentation tasks. However, these neural networks need to be trained on image data before they can be used. There are instance segmentation datasets available for certain tasks. Such as the Cityscapes dataset[13] which has 25000 annotated images for 30 classes. Cityscapes dataset has been used for urban street scene understanding for use in autonomous driving. The 2014 COCO dataset[14] has 2.5 million object instances for 91 object categories. This is a general-purpose dataset with object supercategories such as animal, sports, kitchen, and vehicle.

Annotating these large-scale datasets is time-consuming and expensive. The annotations for instance segmentation have to be made manually. This involves outlining each object in the image and mentioning the name of the object. For the Cityscapes dataset, 1.5 hours were needed to annotate a single image from the 5000 finely annotated images, while the remaining 20000 coarsely annotated images required 7 minutes per image. In the case of the COCO dataset, it took 22 worker hours to annotate every 1000 object instances. That is about 1.32 minutes per annotation. They used Amazon Mechanical Turk[15] to crowdsource the image annotations.

For a small research group or an individual, it is time-consuming to annotate such datasets. Also, there are no ways to estimate the number of images needed to train a model sufficiently. In [16], a neural network model was trained on a manually annotated, 52 image dataset with about 400 objects per image. In the detectron2 tutorial[17], there is an example of a balloon dataset of 61 images with 255 objects being trained to an Average Precision(AP) value of 66. This suggests that for specialized tasks, smaller datasets can be used. However, for generalized tasks such as detecting and segmenting everyday objects in a home setting, large datasets are needed.

Synthetic datasets have been used in [18] to generate datasets using Chroma Key techniques and 3D models. Using 3D models to generate images for computer vision tasks requires models to be prepared by Graphic Designers. Using this method, the annotations can be automated but creating the models requires labour and time.

1.2 Research Goals

The main goal of this thesis is:

Generating synthetic training images for instance segmentation using salient object detection and image compositions

To achieve this goal it is broken down into smaller parts with its own sub-goals to get a perspective on the overarching aim:

- Develop a data generation to create synthetic image compositions with labelled pixel data
  - Take pictures of objects
  - Extract foreground objects from the pictures
  - Generate synthetic images and ground truth labels by making image compositions using the foreground objects
- Experiment with unlabelled foreground objects, noisy images and hybrid datasets
- Train off-the-shelf instance segmentation neural networks using the generated data.
- Evaluate the performance of the neural networks on real, non-synthetic images.
- Evaluate network performance for training with synthetic images on a class not seen in previous datasets.

2 BACKGROUND

2.1 Instance Segmentation

The task of detecting all the instances of a category in an image and marking their pixels was first described by [19]. It was first known as Simultaneous Detection and Segmentation. There has been a significant amount of research on this task since then and has been used in autonomous vehicles and robotics[20]–[22].

2.1.1 Instance Segmentation datasets

The largest dataset for instance segmentation is the Microsoft Common Objects in Context (COCO) [14]. It contains instance-level segmentation of 91 categories over 328k images. The 91 categories in the dataset are chosen based on indoor and outdoor objects that can be recognized by a 4-year-old. Only 80 of these categories are annotated for the instance segmentation task. It is widely used to benchmark neural network models on instance segmentation tasks. The annotation format used in the dataset is used by other datasets as well.

Other instance segmentation datasets are Large Vocabulary Instance Segmentation(LVIS) [23] and Cityscapes[13] . LVIS uses the same images from the COCO dataset but has annotations for around 1000 categories. The annotation format is also the same as the COCO dataset. As mentioned previously, the Cityscapes dataset has instance segmentation annotations for urban scenes from 50 cities around the world.

For robot perception using instance segmentation, there are datasets like Object Cluttered Indoor Dataset (OCID) [24] which contains images of indoor scenes with 89 everyday objects cluttered in them. OCID has images in RGB-D format. The images are captured using two ASUS Xtion Pro cameras. Other such datasets include Autonomous Robot Indoor Dataset (ARID) for Object Detection task and Yale-CMU-Berkeley(YCB) dataset for instance segmentation tasks for robotic manipulation research.

2.2 Evaluation metric for instance segmentation task

The evaluation used for calculating the performance of the network on the validation dataset is the same as the one used for the original Mask R-CNN paper[25]. To determine the accuracy of the model predictions Intersection over Union is used. Intersection over Union is calculated by:

$$\text{IoU} = \frac{\text{model prediction} \cap \text{ground truth}}{\text{model prediction} \cup \text{ground truth}} \tag{2.1}$$

For the COCO metric, IoUs are calculated for 10 different thresholds starting from 0.5 to 0.95 with a step size of 0.05. Precision is the metric which shows the number of correct predicitions made by the model compared to the total number of predictions made by the model. It is calculated using:

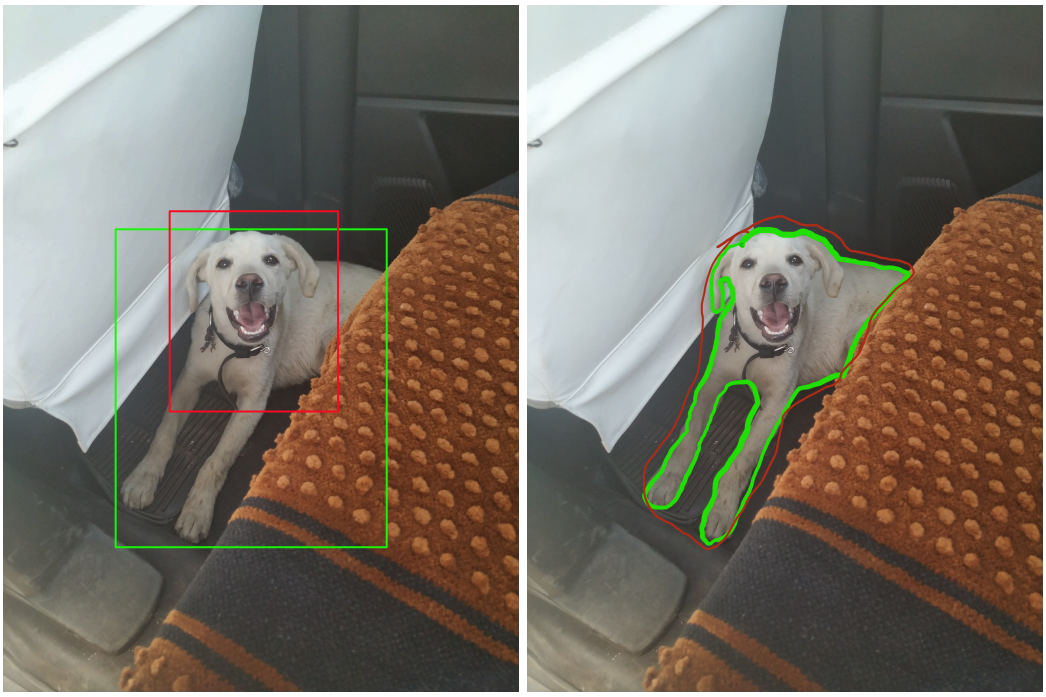
$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.2}$$

Recall shows the completeness of the model predictions with respect to the ground truths in the dataset. Recall is calculated using:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.3}$$

Average Precision is the area under the Precision-Recall curve. AP is calculated for all 10 IoU thresholds and then averaged to get the COCO Average Precision score. Separate metrics for different sizes of the foreground objects also exists. It is called AP over scales in COCO evaluation for detection task. APs is average precision for image objects with area smaller than 32^2 pixels. APl is average precision for image objects with area larger than 96^2 while APm is for image objects between 32^2 pixels and 96^2 pixels.

Average Precision for the instance segmentation task is denoted by AP type 'segm' and the average precision for object detection task is denoted by 'bbox'.



(a) (b)

Figure 2.1: (a) shows ground truth and example model prediction for the object detection task or 'bbox'. (b) shows ground truth and example model prediction for the instance segmentation task or 'segm'. The ground truth is shown in green and model prediciton in red. The bounding box method for localizing objects is not accurate because the dog covers a small part of the image while the rest is background. Segmentation maps are more accurate compared to bounding boxes. The images are only for demonstrating how the metrics are calculated. Both model prediction and ground truth are manually drawn for the images above.

2.3 Instance Segmentation Models

Mask R-CNN is one of the neural networks that has performed well on the Instance Segmentation task[25]. It is a modification of the Faster R-CNN[26] network which was used for object detection tasks. Mask R-CNN models have achieved an AP score of 39.2 on the COCO test dataset. Mask R-CNNs can process images at around 5 frames per second.

Residual networks[27](ResNet) are used as the backbone for Mask R-CNN. ResNets perform better than plain deep neural networks by passing the gradients on some layers forward without performing

any operation. This helps to solve the problem of vanishing gradient where the later layers in the plain deep networks do not perform any learning because the gradients from the loss function are nearly zero. Because of the skipped layers in ResNets, this problem does not occur.

The architecture of Mask R-CNN has two parts - the backbone and the head. The backbone consists of a Residual Neural Network with a feature extractor like Feature Pyramid Network (FPN). The backbone is usually pre-trained with the ImageNet dataset. The backbone learns to detect objects based on low-level features of the image. Pre-training is useful to reduce training times and for limited training data. Although, if there is sufficient training data then using randomly initialized backbone layers leads to similar performance but training time might be slightly higher according to [28].

The head for a Mask R-CNN is the 'Mask predicting head' which produces a bounding box, object classification and the mask of the object. The head is the later layers of the neural network which detect high-level features of the object. Fine-tuning of the model entails training of this head. This head can be replaced to train a different set of categories. This allows for the backbone to be the same for tasks that are similar enough. Fine-tuning of only the head requires significantly less training data than training, both the backbone and head, from scratch.

2.4 Detectron2

Detectron2 [29] is a platform made by Facebook Artificial Intelligence Research (FAIR) group. It is based on the PyTorch library. It allows for easy and quick implementation of neural networks for different tasks like instance segmentation, panoptic segmentation, semantic segmentation, keypoint detection and object detection. Using the configuration file in Detectron2, quick changes can be made to models. It also has pre-trained models of various neural network models.

2.5 Previous works using Synthetic Datasets

To generate synthetic datasets, the first step is to extract foreground objects from images. These extracted foregrounds can then be composited over different backgrounds. In 2008, [18] experimented with synthetic datasets using ChromaKey techniques as well as CAD models for the Object Recognition task. A green screen was used to quickly collect images and extract the foreground objects. They reported that their method of annotating synthetic data was 135 times faster than manually generating the training data. They changed the backgrounds of the images they captured to augment their dataset. Their experiments with using CAD models show that generating synthetic datasets using CAD models requires more time, also adding object shadows did not seem to have any effect on the final performance. The coloured background used has to be lit evenly to produce good foreground extractions. Also, using ChromaKey techniques leads to colour spill in the foreground objects because of reflected light from the background.

In [30], image data was collected for the object classification task using AlexNet [31]. The images were collected using a robotic arm. The robotic arm took pictures of objects on a platform. The robotic arm rotated around the object to take pictures from different angles. This setup allows for quick data capture. They report that capturing 676 images required 580 seconds. The setup used has a green screen which leads to the same problems as discussed above. Using robotic setup allows for faster and automated image capture but the size of the object that can used is limited by the working space of the robot.

In [32], the BigBIRD dataset [33] was used to create synthetic datasets. Each object in the BigBIRD dataset has 600 images taken from different viewpoints. They cut the object instances from the dataset and paste them onto background scenes from the UW scenes dataset [34]. They are trying to solve

the Object Detection task. They use a Faster R-CNN as the detection model. To make the detection network invariant to composition artefacts they used different blending techniques. This functions as a good data augmentation tool but cannot be used for generating synthetic datasets with classes that are not present in any pre-existing datasets. Data augmentation is maximising the effect of an existing dataset. If a dataset does not exist for the given object class, it will have to be manually annotated. Synthetically generating the dataset for the object class could be faster than manually annotating the dataset.

In [35], a Mask R-CNN head with an EfficentNet backbone is used on the COCO and LVIS datasets for instance segmentation task. They report a data augmentation method of using copy-pasting foreground objects onto different background images. They claim an improvement in 2.2 AP over the COCO dataset baseline, using this simple copy-pasting method. This was the only literature for copy-pasting method implemented on instance segmentation for a general purpose task but it is only used as a data augmentation tool. [36] generated synthetic data for seed phenotyping using instance segmentation in agricultural domain.

In [37], Off-the-shelf games are used to generate synthetic data and to annotate. Grand Theft Auto 5 was used to generate a dataset that is similar to Cityscapes. A wrapper was used which allowed them to record, modify and reproduce rendering commands. These commands were then used to generate pixel-level semantic segmentation labels. They report that model trained on their synthetic dataset and 1/3rd real data from CamVid dataset outperformed baseline models trained on CamVid dataset. Generating training data from video games using this method is a lot faster than manually annotating datasets however the object classes are limited by the video games 3D asset library. The same issue also happens with [38] where the SUN-CG dataset[39] is used to generate RGB images using Blender[40]. Capturing images of the objects would take less time than making 3D assets for the object.

2.6 Salient Object Detection, Datasets and Model

As mentioned above, ChromaKey technique provides poor results if the conditions are not right and making 3D models to generate synthetic datasets would be time consuming. For this reason, salient object detection is chosen to extract the foreground objects from images.

Salient object detection task in computer vision deals with distinguishing the most visually distinct objects from a given image. For the given RGB image, a binary mask is produced which segments the salient object. The saliency of the object is determined by the colour, texture, contrast with respect to the background. The binary masks that are produced here can be used to extract foreground objects without the need for Chroma Key techniques.

There have been multiple deep neural networks to solve this problem. Networks that can perform real-time at the same time outperforming other state-of-the-art networks are U^2-Net[41] and BASNet [42] . U^2-Net has a Residual U block architecture while BASNet uses a ResNet-34 architecture.

2.7 Pycococreatortools

Pycococreatortools [43] is an open-source repository for annotating Instance segmentation datasets in the COCO format. It provides various tools for processing binary masks to create a COCO format dataset. It takes the images and binary masks as the input and outputs a JSON file that contains the annotations. The COCO format requires the annotation to be in Run Length Encoding for 'crowd' annotations or Polygon format. The binary masks are converted to the chosen annotation format. Based on the binary mask, other annotation information like bounding box dimensions, width, height ,and area of the segmentation are generated.

3 METHODS

3.1 Proposed Image Synthesis Pipeline

Figure 3.1 shows the different components of the Image Synthesis Pipeline. The name of the component and the function they perform are:

- Foreground Extraction: It can either use ChromaKey methods or U^2 Net to extract the foreground object from Image data provided.
- Image Composer: It takes the extracted foreground objects and performs scaling, rotation, flipping operations on them. It also makes a segmentation map of the composed images which is used for making annotations.
- Image Annotator: It processes the segmentation maps made by Image Composer and outputs a JSON file with annotations like the COCO dataset.
- Neural Network: The Neural Network used here is a Mask R-CNN R-50 FPN. It is implemented in PyTorch[44].

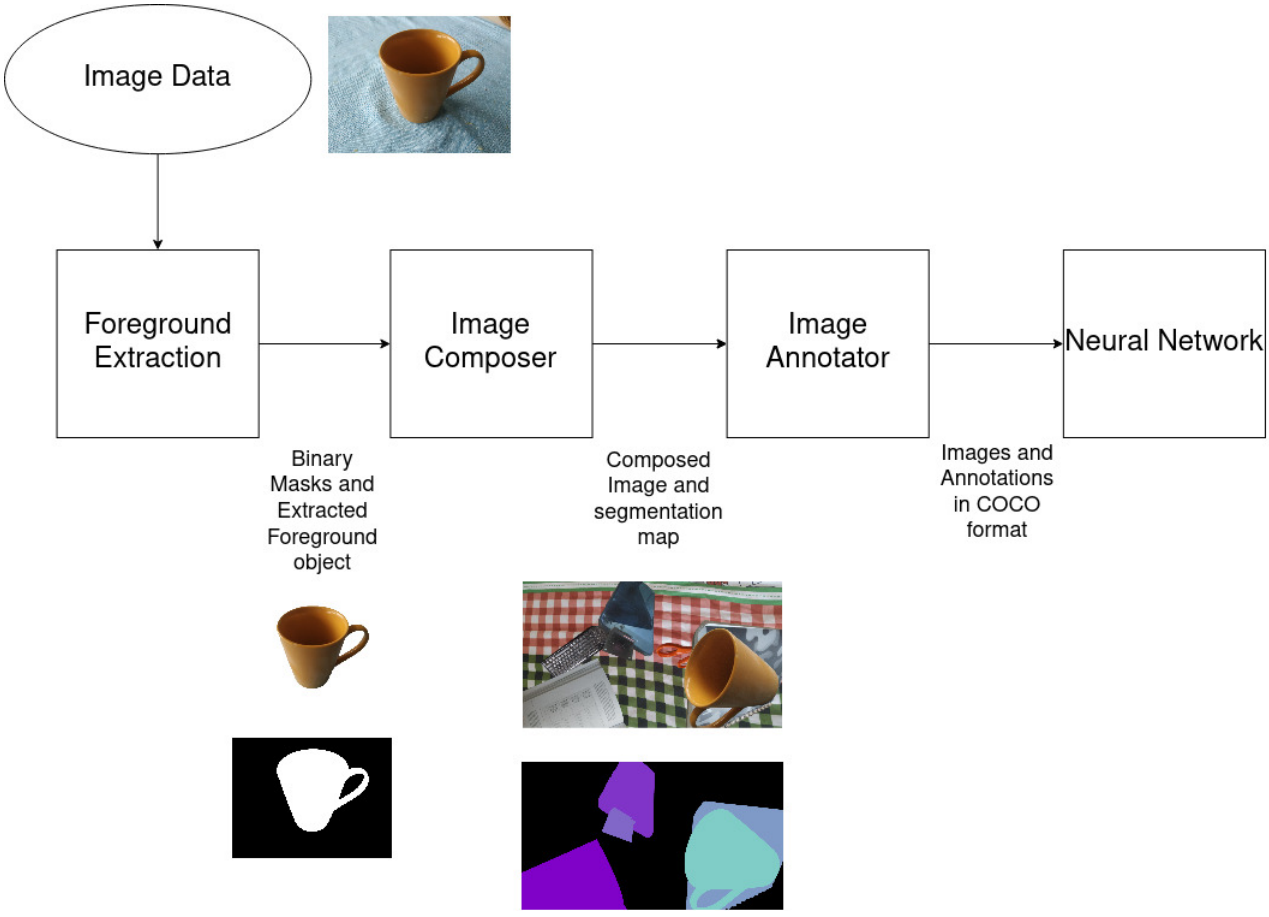


Figure 3.1: Image Synthesis Pipeline

3.1.1 Image Data

Deciding the object categories

Based on the 80 categories available for instance segmentation in the COCO dataset, a list of easily available objects was prepared. This list consisted of 26 objects from the COCO dataset. To test the performance of the image generation pipeline on adding classes not seen in previous datasets, a 'pen' class was added to the list.

Collecting Images

From the aforementioned list, all the unique objects for each category were collected. These objects were photographed in front of a contrasting background. This was done to help the salient object detection network to make segmentation easier. The objects were photographed from different angles and also changing the pose of the objects. At the end of this step, 1092 images were collected for the 26 objects that appear in the COCO dataset. For the 'pen' category 226 images were collected. The images were taken using a Mi A2 smartphone camera. Image resolution was 4000 × 3000. Total time needed to collect all the images was around 4 hours. Number of images taken for each class and number of instances appearing in the COCO dataset is shown in Table 3.1.

3.1.2 Foreground Extraction with Salient Object Detection

The salient object detection network used was U^2-Net [41]. U^2-Net provided binary masks for the images in real-time. This is important when batch processing thousands of images. However, one drawback is that it produces the masks at a resolution of 320 × 320 pixels. This output was enlarged to the original resolution of the images. This caused the edges of the foreground object to become blurred. To refine the foreground object boundary, CascadePSP[45] was used and then the foreground objects segmentation mask was binarized.

3.1.3 Image Composer

After the foregrounds objects had been extracted, these were used by the Image composer script to generate image compositions. The classes to include in the image compositions were selected by adding the name of the object in the 'chosen_objects' list or if all the objects were to be included then the 'choice' variable can be changed as required. The number of objects to include in each image were set using the variables 'upper_lim' and 'lower_lim'. The number of images to generate can be prescribed by changing the value of the 'BatchSize' variable. The resolution of the Image Compositions can be set using the variable 'bg_size'. The image composer script uses multiprocessing to reduce the time required to generate the images.

The foreground objects were placed randomly on the backgrounds provided. The size of the foreground was randomly selected between 0.05 to 0.75 times the width of the image. The foregrounds were also randomly flipped. Also, complete rotation of the foregrounds was possible and the angle was chosen at random. This was done to augment the few images that were available. It is possible for the images to be only be partially visible, also overlapping foreground objects can also be present in the synthetic dataset to mimic real images.

The Image composer script produces three outputs. The first is the image compositions, second is the coloured masks which is used to generate the binary annotation masks.The binary masks are the third output of the Image Composer. The synthetic images had a resolution of 1920 × 1080. A Gaussian filter of 1 pixel radius is applied on all the images. Number of objects in each image was randomly selected between 2 and 10.

Classes	Images collected	Number of instances in COCO dataset
Apple	21	1662
Backpack	36	5756
Banana	33	2346
Book	205	5562
Bottle	110	8880
Carrot	18	1764
Cell phone	60	5017
Chair	12	13354
Clock	23	4863
Cup	161	9579
Fork	30	3710
Handbag	7	7133
Keyboard	24	2221
Knife	61	4507
Laptop	37	3707
Mouse	23	1964
Orange	10	1784
Potted Plant	8	4624
Remote	20	3221
Scissors	28	975
Spoon	58	3682
Teddy Bear	9	2234
Toothbrush	59	1041
Umbrella	11	4142
Vase	18	3730
Wine glass	10	2643
Total	1092	110101

Table 3.1: Table showing number of images collected and number of instances appearing in the COCO dataset for the given class.The classes highlighted in red are selected in the ten class dataset.

3.1.4 Image Annotator

The inputs to the Image Annotator are the binary masks produced from the segmentation map. The annotations are made in the format of the COCO dataset. The annotations are stored in the JSON file format. This annotation file is then used to train the neural network. This annotation file can be used in any Deep Learning framework to train instance segmentation neural networks. The script for annotating the dataset according to the COCO format was taken from Pycococreatortools [43].

For the object detection task, a dataset in COCO format requires the following information:

- Info
- Licenses
- Categories
- Images
- Annotations

The classes which have to be annotated can be added to the 'CATEGORIES' list in the Image Annotator script. The COCO dataset format requires the category id here to begin at 1, 0 is reserved for the 'background'. Pycococreatortools converts the binary masks into polygon notation which is stored in the 'Annotations' list along with other information generated from the polygons such as area, bounding box dimensions and the category id of the object and the image id in which the object appears.

3.1.5 Neural Network

The Neural Network selected is Mask R-CNN based on ResNet-50 architecture and Feature Pyramid Network. There are Mask R-CNN models that are deeper than 50 layers and have better mask AP values but these take longer to train and run inferences on image. Also, deeper models tend to be take up more space on the GPU. The chosen model is a good trade-off between speed of training and inference, model size and accuracy. The neural network models were trained on a Nvidia Titan Xp GPU.

3.2 Folder Structure for the Synthetic Image Generation Pipeline

```
Base
|-- Annotations
|-- Backgrounds
|-- Classes
|   |-- apple
|   |-- backpack
|   |-- ...
|   `-- wine glass
|-- ColouredMasks
|-- Composed
|-- EFObjects
|   |-- apple
|   |-- backpack
|   |-- ...
|   `-- wine glass
`-- Mask
    |-- apple
    |-- backpack
```

```
|-- ...
`-- wine glass
```

- Annotations: It contains the binary masks which are used for annotations. The annotation masks are saved in the format of 'ImgNumber_ObjectName_InstanceNum.png'.
- Backgrounds: The backgrounds used by the image composer are stored in this folder.
- Classes: The raw image data is stored in sub-folders here.
- ColouredMasks: It contains the coloured masks from which the binary masks are made. Coloured mask is saved with the same name as the composed image.
- Composed: The composed images are stored in this folder.
- EFObjects: Extracted foreground objects are stored in this folder. It follows the same sub-folders as "Classes".
- Mask: It contains the binary masks made by the foreground extraction method. They are used when composing the images. It also has the same sub-folders as "Classes"

3.3 Experiments

3.3.1 Single class dataset

To check the performance of a neural network that is trained on synthetic images, a toy dataset consisting of 61 synthetic images was created. This dataset contains 269 instances of 'pen' object. The dataset was modelled after the example shown in the Detectron2 tutorial on Google Collaboratory [46]. The example has 61 images and 255 instances of 'balloon' object. While selecting the backgrounds for the synthetic dataset, care was taken not to include a pen in the background because having an unlabelled pen instance in the background would probably confuse the neural network and lead to poor performance.

The training parameters for the Mask R-CNN model are mentioned below:

- Learning Rate: 0.00025
- Images per batch for Backbone: 2
- Batch size per image for Region of Interest head: 128

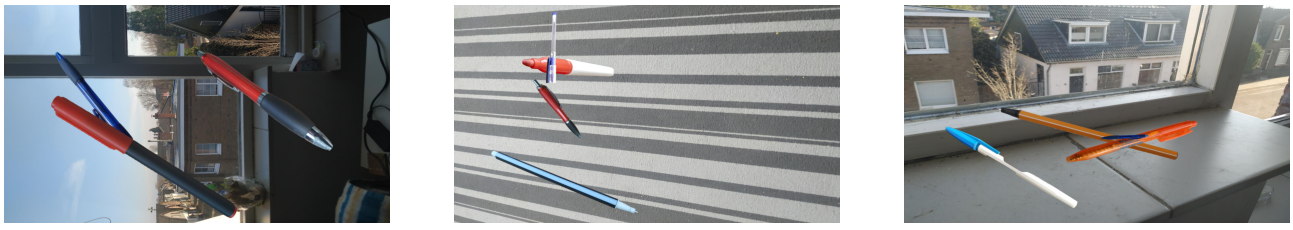


Figure 3.2: Examples from the synthetic Pen dataset

3.3.2 Two-class dataset

A two-class dataset consisting of 'book' and 'bottle' classes was created. The two classes were chosen because these two classes had the highest number of images collected as can be seen in Table 3.1. It consists of 200 synthetic images with 862 instances of which 441 belong to 'book' while 421 belong to 'bottle'. This dataset also contains unlabelled instances of foreground objects. This was done to make

sure that the neural network is invariant to the artefacts from copy-pasting. Some example images are shown in Figure 3.3 The training parameters for the Mask R-CNN model are mentioned below:

Figure 3.3: Examples from the two class synthetic dataset. Annotated instances are highlighted.

- Learning Rate: 0.00025
- Images per batch for Backbone: 2
- Batch size per image for Region of Interest head: 256

3.4 Ten classes datasets

The training parameters for the Mask R-CNN model are mentioned below:

- Learning Rate: 0.00025
- Images per batch for Backbone: 2
- Batch size per image for Region of Interest head: 512

3.4.1 Basic Synthetic Dataset

This dataset contains 10 classes with the highest number of images as shown in Table 3.1. It has 5999 images and 26016 object instances. It only has these 10 classes in the images. No unlabelled foreground classes were added to this dataset. A Gaussian filter was used at the end of the image composition step to filter image composition artefacts. It was observed that the Image Annotator added some wrong annotations due to salt and pepper noise in the binary masks. This caused the bounding boxes made by the annotator to be bigger than the actual foreground object. This was caused by a poor binarization threshold in the step where binary masks were created. Also, sometimes foreground objects that did not exist in the image would get annotated. This was a bug from Pycococreatortools and it has been fixed for the other datasets. Examples of this can be seen in Figure 3.4.

Two datasets of Basic type were made. The first dataset has 26016 object instances while the second dataset contains 31841 object instances. This was done to make the dataset similar to the subset of the COCO dataset used for training which is used as the benchmark dataset.

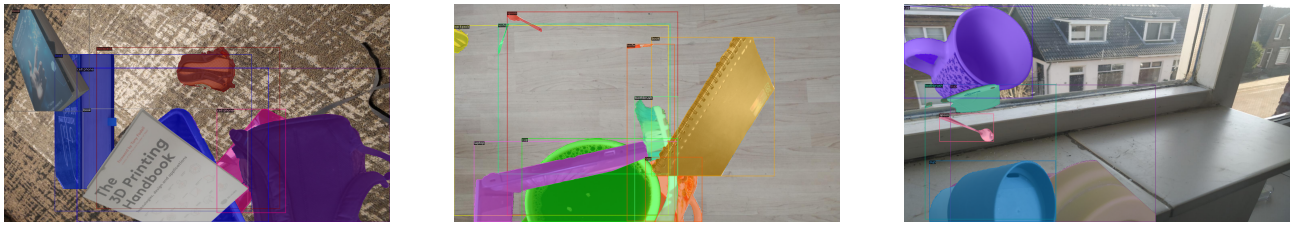


Figure 3.4: Examples from the synthetic 10 classes dataset. The instances for spoon and knife in the center image have bounding boxes much larger than the actual object. This was caused by a bug in the annotation step. This was fixed for the subsequent datasets.

3.4.2 Synthetic Dataset with Unlabelled instances

This dataset contains the same 10 classes as shown in Table 3.1. It also has 5999 images and 28771 object instances. It has unlabelled foreground objects added. This is similar to the foreground-shaped cut-outs implemented in [47]. This is done to make the neural network model invariant to the composition artefacts. Examples from the dataset are shown in Figure 3.5

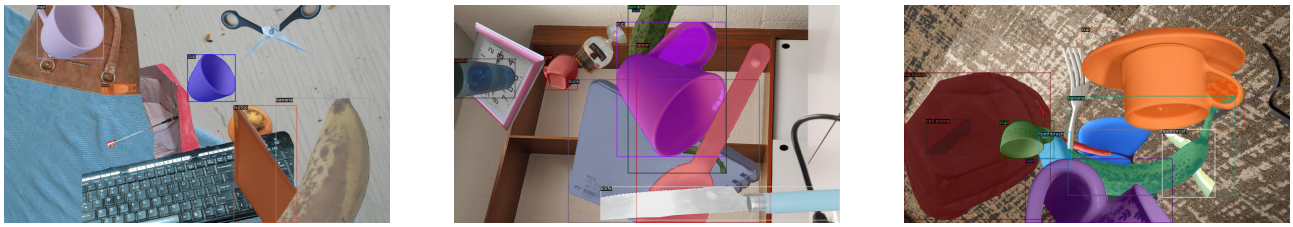


Figure 3.5: Examples from the synthetic 10 classes dataset with unlabelled instances. Unlabelled instances can be seen which are not highlighted.

3.4.3 Hybrid Datasets

Hybrid datasets consisting of Synthetic images as well as real images from the COCO dataset were created. This was done to test the performance of the neural network if it was trained on both synthetic datasets and real images. The idea behind this is that the easily produced synthetic images would teach object semantics i.e shape, texture, colour while the real images would teach the syntax i.e contextual placement of the objects.

Four hybrid datasets were made with different amounts of synthetic and real images.

- Hybrid50S50R contains 6000 images of which 3000 are synthetic and 3000 are real images with 33561 object instances.
- Hybrid75S25R contains 6000 images of which 4500 are synthetic and 1500 are real images with 29409 object instances.
- Hybrid90S10R contains 6000 images of which 5400 are synthetic and 600 are real images with 27798 object instances.
- Hybrid98S02R contains 6000 images of which 5880 are synthetic and 120 are real images with 27311 object instances.

The number of object instances was randomly determined for each image. Only the upper and lower limits for those were provided.

3.4.4 Subsets of the Hybrid datasets

Models were also trained with real images from the Hybrid datasets to check how well models trained on this part of the dataset perform on the validation dataset. These have been named COCO_3000, COCO_1500, COCO_600 and COCO_120. Synthetic datasets of the same size and similar number of instances were also created.

3.4.5 Datasets with Noise

After looking at the results for the models on the datasets mentioned above, the hybrid datasets and synthetic subsets had noise added to observe the effect of noise addition. Since Real-world images contain various types of noise due to factors like poor lighting or blur, adding noise to training image helps the model to become robust to these effects. Zero mean Gaussian noise with standard deviation of 10 was added to all the images.

The noise injected datasets are:

- Hybrid50S50Rnoise
- Hybrid75S25Rnoise
- Hybrid90S10Rnoise
- Hybrid98S02Rnoise
- Synth3000noise
- Synth1500noise
- Synth600noise
- Synth120noise

3.4.6 Retrained Models

The models trained on the Synth_120noise dataset and the Hybrid98S02Rnoise dataset were retrained on the COCO_120 dataset. This was done to see how well a retrained model performs compared to a model trained on a single dataset.

The training parameters for the Mask R-CNN model are mentioned below:

- Learning Rate: 0.00025
- Images per batch for Backbone: 2
- Batch size per image for Region of Interest head: 512

4 RESULTS

4.1 Foreground Extraction

The total time needed for foreground extraction from the 1318 images using U^2-Net was 2 hours 28 minutes or 6.5 seconds per image. Five seconds are required to refine the segmentation mask than generating the segmentation mask for each image. The extracted foreground objects are stored in "Base/EFObjects/ObjectName".

Figure 4.1: Foreground extracted with U^2-Net. Edges of the extracted foreground have some colour from the background

(a) (b) (c) (d)

Figure 4.2: Steps taken for foreground extraction. (a) shows the image that is fed to U^2-Net. (b) is the output from the U^2-Net. The edges of the foreground object because of the resizing from the output of U^2-Net. (c) shows the image after binary thresholding. This removes the blurry edges but it also removes the some parts of the foreground. (d) shows the output from the CascadePSP network.

4.2 Image Composer

The time needed to generate 6000 images using this Image Composer script was 52 minutes or 0.52 seconds per image. The reported time is for a 40-core Intel Xeon E5-2630 processor. The time needed to generate 6000 images on a single core was 11 hours and 21 minutes or 6.8 seconds per image. Examples of these outputs are shown in Figure 4.3 and Figure 4.4.

Figure 4.3: The generated image and its corresponding coloured mask.

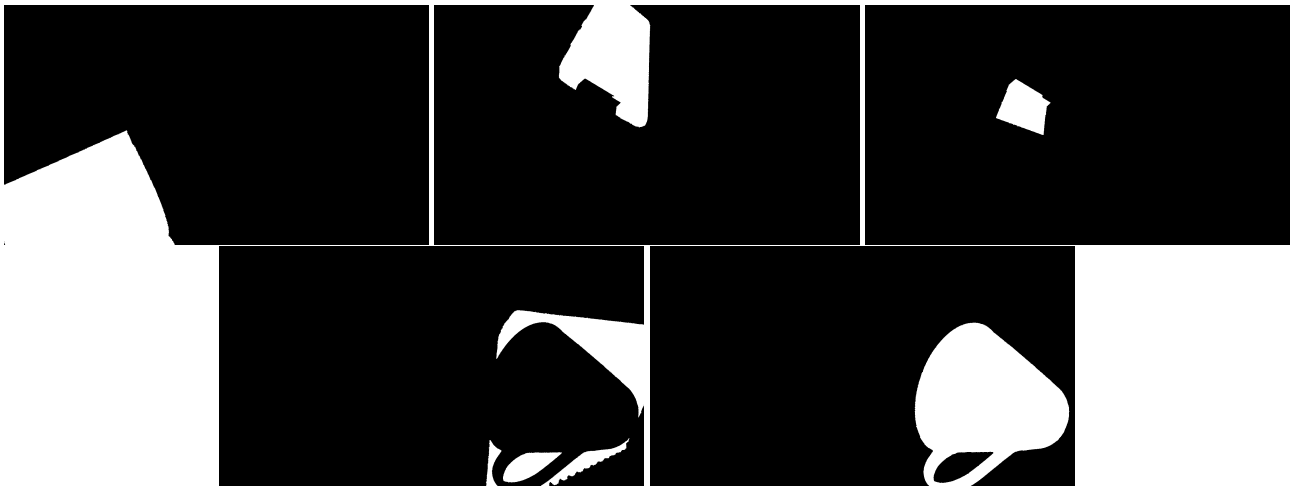


Figure 4.4: The generated binary masks from the coloured mask. These are used for making the annotations

4.3 Single Class Dataset

A validation set consisting of 16 real images was created using images downloaded from the internet. The validation set has 31 instances of 'pen' object. These images were collected from the internet to make sure they were not the same pens as in the Synthetic dataset. The Time needed to collect and annotate these images for the validation dataset was 29 minutes. The open-source VGG Image Annotation[48] tool was used for annotating the dataset. The model trained on the dataset of 61 synthetic images was tested on this validation dataset. Table 4.1 shows that the training models using the generated synthetic images is possible and some model predictions for the validation set are shown in Figure 4.5.

Model name	**APtype**	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**
SynthPen	bbox	57.061	77.647	67.909	40	40.594	61.777
	segm	41.471	69.002	44.387	16	9.924	51.55

Table 4.1: Test results on the validation dataset for the model trained on synthetic pen dataset. The bbox denotes the bounding box test for the object detection task and segm denotes segmentation test for instance segmentation test. AP50 and AP75 refer to Average Precision calculated at 0.5 and 0.75 IoU thresholds respectively. APs is average precision for image objects with area smaller than 32^2 pixels. APl is average precision for image objects with area larger than 96^2 pixels while APm sits between APs and APl.

4.4 Two Class Dataset

The validation set for the two-class dataset is a subset of the COCO validation dataset from 2017. This subset contains 200 images. To compare the model trained on the Synthetic dataset, another

Figure 4.5: Examples of the predictions made by the model trained on the Synthetic Pen dataset.

model was trained on a 200 image subset of the COCO 2017 train dataset. The validation dataset contains 200 images with 430 instances of 'bottle' object and 422 instances of 'book' object. Results for the models trained on the Synthetic dataset and COCO subset are shown in Table 4.2. Figure 4.6 shows the performance for each class as well as overall AP for both models. Since, the model trained on synthetic images does not see humans during training, the model overgeneralizes them as 'bottle'.

Model name	**APtype**	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**	**APbottle**	**APbook**
Synthetic	segm	11.234	17.994	12.797	6.739	23.05	29.738	19.028	3.44
COCO_subset	segm	16.792	29.938	17.049	11.853	29.858	33.424	27.76	5.825
Synthetic	bbox	12.042	18.916	13.615	8.548	26.411	27.86	20.122	3.963
COCO_subset	bbox	20.114	34.722	20.857	16.974	30.938	28.682	31.335	8.892

Table 4.2: Results for the models trained on two class dataset when tested on COCO subset

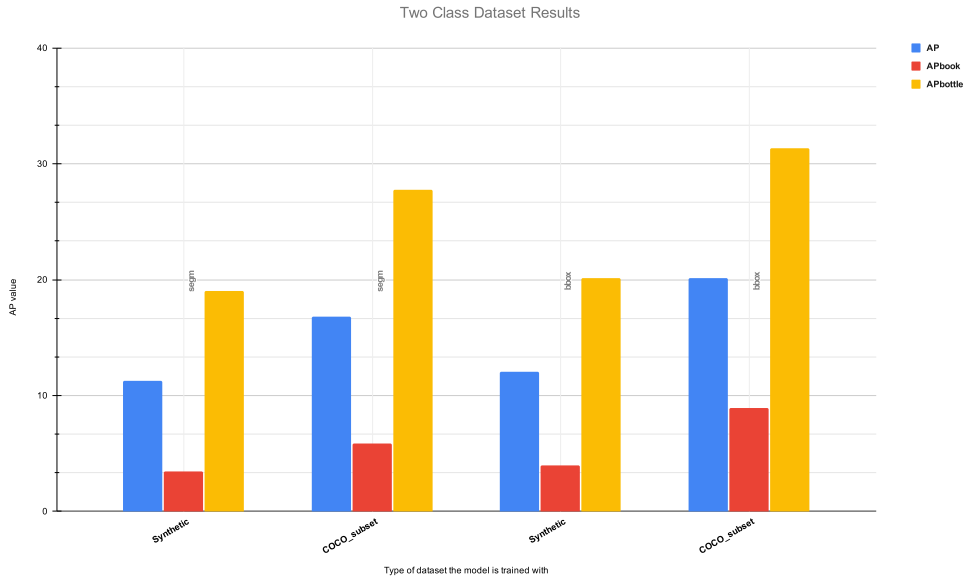


Figure 4.6: The model trained on Synthetic images performs poorly compared to model trained on the COCO subset.

4.5 Ten Class Datasets

The benchmark for the Ten class dataset was also a subset of the COCO validation dataset from 2017. The validation dataset contains 1399 images out of the 5000 images present in the COCO validation

dataset. This validation set is used to check the performance of all the models trained on a dataset that has these ten classes.

4.5.1 Basic Datasets

Table 4.3 shows the AP scores for the models trained on Synthetic, Real and Hybrid images. For the given ten classes, the models trained on the synthetic dataset perform poorly compared to the model trained on real data, see Tabel 4.3. Also, adding unlabelled object instances in the background of the synthetic images provides an improvement in the AP scores. The models trained on hybrid models have performance that is similar (for Hybrid50S50R and Hybrid75S25R) or worse (for Hybrid 90S10R and Hybrid98S02R) than the models trained on the real image subsets of these datasets. The results are shown in Figure 4.7.

Model Name	**Trained on**	**AP Segm**	**AP bbox**
COCO10cls_retrial	Real	22.359	25.829
Hybrid50S50R	Hybrid	20.647	23.786
COCO_3000	Real	20.372	23.080
COCO_1500	Real	19.299	20.988
Hybrid75S25R	Hybrid	19.098	21.782
COCO_600	Real	19.046	20.983
COCO_120	Real	16.408	16.903
Hybrid90S10R	Hybrid	16.336	18.647
Hybrid98S02R	Hybrid	11.521	13.0490
SynthUL10_30k	Synthetic	6.780	7.441
Synth10_new	Synthetic	4.384	4.502

Table 4.3: Results of the models trained on the basic Synthetic, Hybrid and Real Datasets. Models trained on real images perform better than models trained on synthetic datasets. While hybrid datasets perform better than synthetic datasets their performance is similar to their real image subsets, it would be better to train the models only on the real image subsets.

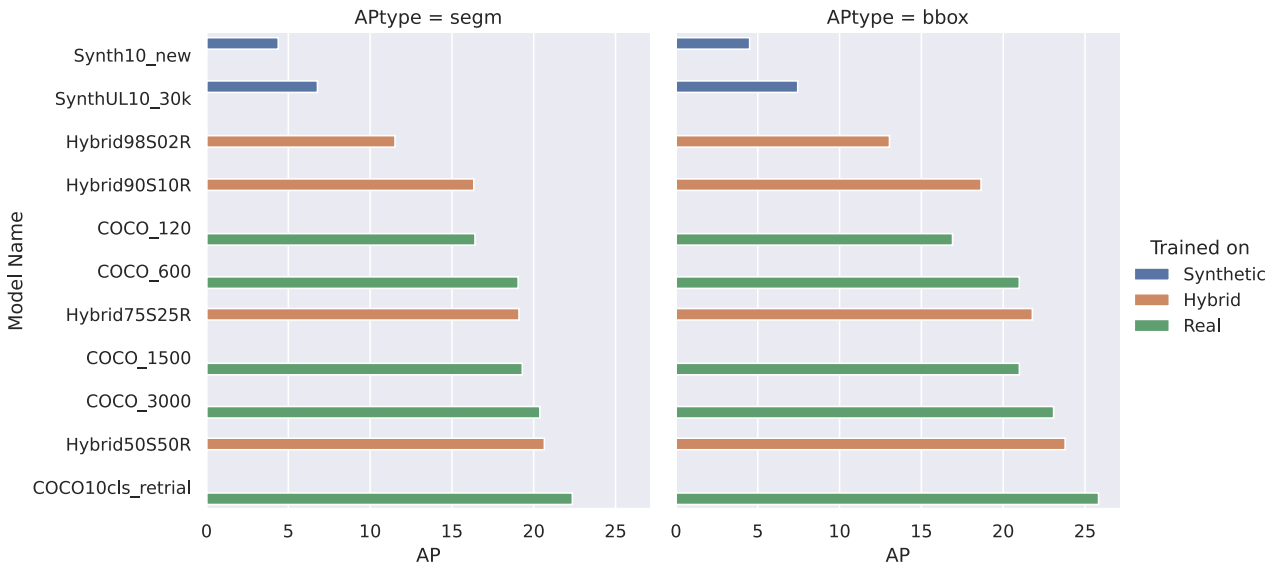


Figure 4.7: Results of the models trained on the basic synthetic(blue), hybrid(yellow) and real(green) datasets. The results for instance segmentation task are shown in 'segm' and object detection task are shown in 'bbox'

4.5.2 Datasets with Noise

The results for the effects of noise are shown in Table 4.4. Adding Gaussian noise to the synthetic images has a positive effect on the AP scores. However, for the models trained on hybrid datasets, there is a drop in performance compared to the same datasets without noise added. The results are shown in Figure 4.8.

Model Name	**Trained on**	**AP Segm**	**AP bbox**
Synth_3000	Synthetic	4.393	4.809
Synth_1500	Synthetic	6.531	7.251
Synth_3000noise	Synthetic_noise	6.820	7.666
Synth_600	Synthetic	8.048	8.710
Synth_1500noise	Synthetic_noise	8.872	9.488
Synth_120	Synthetic	10.090	10.693
Synth_600noise	Synthetic_noise	10.145	11.120
Synth_120noise	Synthetic_noise	10.853	11.427
Hybrid98S02Rnoise	Hybrid_noise	11.454	12.554
Hybrid98S02R	Hybrid	11.521	13.049
Hybrid90S10Rnoise	Hybrid_noise	15.863	17.816
Hybrid90S10R	Hybrid	16.336	18.647
Hybrid75S25Rnoise	Hybrid_noise	17.813	20.444
Hybrid75S25R	Hybrid	19.098	21.782
Hybrid50S50Rnoise	Hybrid_noise	20.005	22.897
Hybrid50S50R	Hybrid	20.647	23.786

Table 4.4: Results of models trained on synthetic and hybrid datasets with injected Gaussian noise.

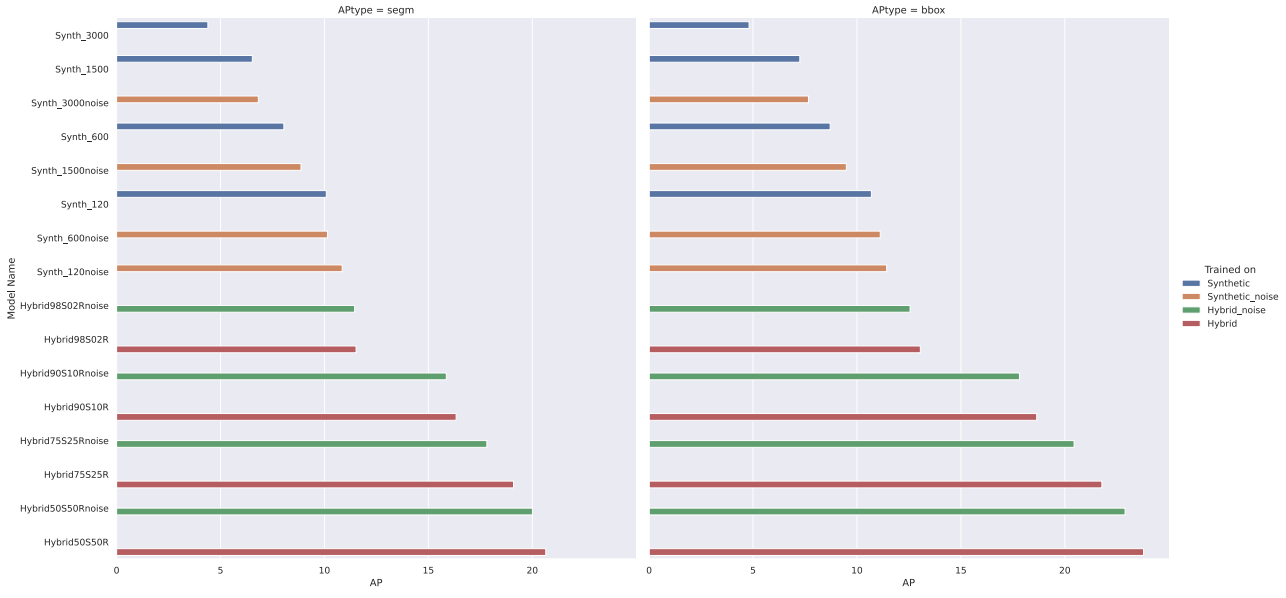


Figure 4.8: Results of models trained on datasets trained on noise injected Synthetic and Hybrid datasets.The results for instance segmentation task are shown in 'segm' and object detection task are shown in 'bbox'

4.5.3 Retrained model

The results for the retrained model are shown in Table 4.5. The model trained on the Synth_120noise and COCO_120 performs better than models trained on only one of the datasets. The results are shown in Figure 4.9.

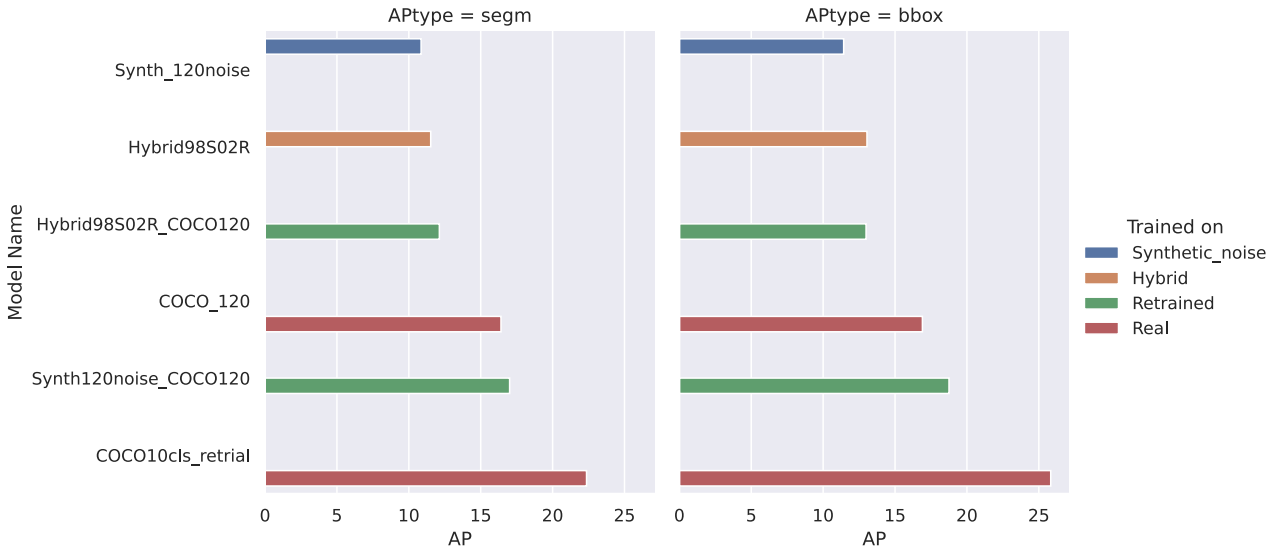


Figure 4.9: Results of the retrained model with AP values for the models trained only the individual datasets. The results for instance segmentation task are shown in 'segm' and object detection task are shown in 'bbox'

Model Name	**Trained on**	**AP segm**	**AP bbox**
Synth_120noise	Synthetic_noise	10.853	11.427
Hybrid98S02R	Hybrid	11.521	13.049
Hybrid98S02R_COCO120	Retrained	12.123	12.986
COCO_120	Real	16.408	16.903
Synth120noise_COCO120	Retrained	17.010	18.753
COCO10cls_retrial	Real	22.359	25.829

Table 4.5: Results of the retrained models and the models trained only on a single dataset.

Model Name	**Trained on**	**AP Segm**	**AP bbox**
SynthUL10_30k	Synthetic	6.780	7.441
Synth_120noise	Synthetic_noise	10.853	11.427
COCO_120	Real	16.408	16.903
Synth120noise_COCO120	Retrained	17.010	18.753
Hybrid50S50R	Hybrid	20.647	23.786
COCO10cls_retrial	Real	22.359	25.829

Table 4.6: Results for the best performing models for each type of dataset that it was trained on.

Model Name	**Trained on**	**AP Segm**	**AP bbox**
COCO10cls_retrial	Real	22.3599230126633	25.8290412509275
Hybrid50S50R	Hybrid	20.6473487944007	23.7862924613006
COCO_3000	Real	20.3720406108857	23.0804160673462
COCO_sub_30k	Real	20.0932892360644	22.7893008635648
Coco_sub_25k	Real	20.0754779167853	21.3652843390615
Hybrid50S50Rnoise	Hybrid_noise	20.0057397609868	22.897084027899
COCO_1500	Real	19.2998030866941	20.9880890942207
Hybrid75S25R	Hybrid	19.0985588346187	21.7823048961323
COCO_600	Real	19.0466465664014	20.9839608817647
Hybrid75S25Rnoise	Hybrid_noise	17.81391414654	20.4443130004371
Synth120noise_COCO120	Retrained	17.0107642810354	18.7538981444623
COCO_120	Real	16.4089314925237	16.9035177186389
Hybrid90S10R	Hybrid	16.33617043679	18.6477623775332
Hybrid90S10Rnoise	Hybrid_noise	15.8634721450247	17.8167799764698
Hybrid98S02R_COCO120	Retrained	12.1233482391871	12.986023392189
Hybrid98S02R	Hybrid	11.5219484350429	13.0490318285698
Hybrid98S02Rnoise	Hybrid_noise	11.4549652507704	12.5546001858221
Synth_120noise	Synthetic_noise	10.8538563602302	11.4277551557503
Synth_600noise	Synthetic_noise	10.1456286406607	11.1201775462215
Synth_120	Synthetic	10.0909362173562	10.6932368072078
Synth_1500noise	Synthetic_noise	8.87256809267108	9.48806213290685
Synth_600	Synthetic	8.04887874640627	8.71078538572301
Synth_3000noise	Synthetic_noise	6.82066300466192	7.66642831036009
SynthUL10_30k	Synthetic	6.78003565748531	7.44138373823832
Synth_1500	Synthetic	6.53164701712607	7.25121175367227
SynthUL10_1k	Synthetic	6.47045917324177	7.48715856506703
SynthUL_10cls_retrialSTEPS	Synthetic	6.06917487661886	6.81542416253592
Synth10_retrialwSTEPS	Synthetic	4.93368671857218	5.21580851557049
Synth_3000	Synthetic	4.3932626842535	4.80973619384677
Synth10_new	Synthetic	4.38482643859574	4.50278792036657
Synth10	Synthetic	3.75106635787171	3.97694824885489

Table 4.7: AP scores for all the models that were trained.

5 DISCUSSION

All of the ten-class datasets have a low value of AP compared to the 37 AP of the pretrained Mask R-CNN model provided by Detectron2. The reason for this is that the chosen classes have a poor AP score on the pre-trained Mask R-CNN as well, this is shown in Table 5.1. The average AP for these 10 classes is 25.88. Also, the model that provided the best AP among the ten classes datasets was COCO10cls_retrial with an AP of 22.359. This is close enough to the value reached by the model trained on the full dataset which has around 32000 images for these ten classes alone.

The models trained on synthetic data tend to perform poorly as compared to the models trained on real data. The reason for this could be that models trained on synthetic datasets tend to overfit on image composition artefacts. This was also observed by [32], [47]. To avoid this overfitting, they suggested training the network with copies of the image with different filters applied. Only Gaussian filtering was performed on the synthetic images that were produced. To avoid overfitting to these artefacts, style transfer techniques like [49] could also be used.

It is also observed that models trained on synthetic with unlabelled instances in the background perform better than models trained on the basic synthetic image dataset. This suggests that adding those unlabelled instances helps the model become invariant to composition artefacts. Since, the models trained on synthetic datasets did not have a 'person' class, they tended to misclassify humans as 'bottle' or 'backpack'. If humans are introduced to the synthetic image as unlabelled in the background or labelled objects this error could be reduced.

Hybrid datasets did not perform better than their real image subsets. Only Hybrid50S50R model scored better than its subset, COCO_3000. The rest of the models trained on Hybrid datasets performed poorly compared to their subset, particularly Hybrid98S02R. Using these hybrid datasets provides no benefit to the model performance.

It can be seen from the results in 4.4 that addition of noise to the synthetic datasets improves the AP score for the model considerably. However, this is not seen in hybrid datasets. There is a loss of AP score for all the models that are trained on noise injected Hybrid datasets.

The retrained models perform better than the models trained only on the single dataset. It can be seen that there is a significant improvement in the performance of the Synth_120noise model and the retrained model. It also performs slightly better than the COCO_120 model. The Synth120noise_COCO120 model is trained only on 240 images.

This work only considered randomly placing the the foreground objects in the image, however real images have contextual information. Generating images with contextual awareness could help improve the performance of models trained on synthetic datasets. [47], [50], [51] used Generative Adversarial Networks, Probability Guided heat maps and neural networks to predict placement of foreground objects for contextually accurate synthetic images.

category	AP	category	AP	category	AP
person	47. 659	bicycle	17. 969	car	41.815
motorcycle	32. 986	airplane	49.252	bus	63. 667
train	61. 038	truck	35. 068	boat	23. 022
traffic light	26. 765	fire hydrant	62. 378	stop sign	66. 174
parking meter	45.015	bench	17.275	bird	30. 338
cat	66. 854	dog	57. 179	horse	41.555
sheep	43. 681	cow	46. 896	elephant	55. 802
bear	69.355	zebra	56.278	giraffe	51.522
backpack	16.44	umbrella	44.65	handbag	14. 933
tie	31.189	suitcase	39.446	frisbee	62.388
skis	3.222	snowboard	21.955	sports ball	46. 843
kite	30.874	baseball bat	24.689	baseball glove	38.559
skateboard	31.492	surfboard	31.171	tennis racket	53.682
bottle	38.238	wine glass	30.948	cup	41.307
fork	15.283	knife	12.811	spoon	12.155
bowl	40.012	banana	19.467	apple	20.167
sandwich	36.739	orange	30.311	broccoli	21.661
carrot	18.588	hot dog	27.428	pizza	50. 275
donut	45.267	cake	35.038	chair	18.14
couch	36.135	potted plant	22.723	bed	32.042
dining table	16.106	toilet	57.366	tv	57.325
laptop	59.19	mouse	64.251	remote	28.451
keyboard	50.812	cell phone	34.009	microwave	55.995
oven	31.137	toaster	43.311	sink	35.765
refrigerator	57	book	10. 240	clock	50.323
vase	36.551	scissors	20.594	teddy bear	43.648
hair drier	0.636	toothbrush	14.988		

Table 5.1: Result of the Mask R-CNN ResNet-50 FPN model provided by Detectron2 on COCO val2017 dataset. The selected ten classes are highlighted.

Based on the results, it can be concluded that the synthetic dataset generation pipeline can be used to train instance segmentation models. The best models trained on synthetic datasets achieves about 50% AP as a model trained on a real dataset. Generating synthetic datasets required only 20% of the time required to manually annotating a dataset. Annotating images manually requires constant attention from the user. For the synthetic image generation pipeline, user is required only for the image collection step and the rest of the steps are executed by the computer.

It was also observed that smaller synthetic datasets (<600 images) had better results than larger datasets. Adding unlabelled objects in the background could be a good way to improve the performance of the instance segmentation model. One way to do this would be to check the misclassifications and then adding unlabelled instances accordingly.

The recommended way to generate datasets would be to initialize model training with the synthetic dataset and then use the model predictions to improve the annotations for the rest of the raw real image data. Initializing from synthetic datasets provides the benefit of having a model that does perform well on at least the AP-large (instances with an area larger than 96^2) and AP-medium instances(instances with an area between 32^2 and 96^2). The focus can then be shifted to annotating only the AP-small (area smaller than 32^2) instances. The AP-small comprise 41% of the total COCO dataset [14]. Having a synthetic dataset that is similar to the actual validation set in this regard could increase the accuracy of the instance segmentation model. The total number of object instances for the datasets and number of instances across different scales is shown in table 5.2.

Dataset Name	Total number of instances	Area-small	Area-medium	Area-large
COCO train2017	860001	356338	295160	208498
COCO val2017	36781	15315	12569	8897
COCO_120	813	444	283	85
COCO_1500	9633	5050	3607	974
COCO_3000	20575	11211	7397	1960
COCO_600	4170	2170	1612	388
COCO_train_subset	33411	17314	12391	3698
COCO_val_subset	4963	2826	1589	547
COCO2cls_train	862	564	253	45
COCO2cls_val	852	603	212	37
hybrid50S50R	33561	11486	9025	13043
hybrid75S25R	29409	5514	6072	17820
hybrid90S10R	27798	2686	4614	20498
hybrid98S02R	27311	1440	3682	22186
pen_train	269	18	55	196
penVal	31	1	4	26
Synth_10cls	26016	540	3103	22372
Synth_120	539	15	62	462
Synth_600	3201	83	440	2678
Synth_UL10cls_new	31819	787	4164	26866
Synth10cls_new	31841	768	4071	27001
Synth2clsUL	833	10	85	738

Table 5.2: Table showing the number of object instances in each dataset. Area-small refers to object instances having an area under 32^2 pixels. Area-large is for object instances with an area above 96^2 pixels. Area-medium is for object instances with an area between 32^2 and 96^2 pixels. Real image datasets have more object instances for the Area-small and Area-medium scale compared to the synthetic and hybrid datasets. Majority of the images in the real datasets are Area-small.

Bibliography

[1] T S Huang. "Computer Vision: Evolution and Promise". en. In: (), p. 5.

[2] Marjolein Bruijning, Marco D. Visser, Caspar A. Hallmann, et al. "trackdem: Automated particle tracking to obtain population counts and size distributions from videos in r". en. In: *Methods in Ecology and Evolution* 9.4 (2018). _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.12975, pp. 965–973. ISSN: 2041-210X. DOI: https://doi.org/10.1111/2041-210X.12975. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.12975 (visited on 05/17/2021).

[3] Qing Li, Gaochen Min, Peng Chen, et al. "Computer vision-based techniques and path planning strategy in a slope monitoring system using unmanned aerial vehicle". en. In: *International Journal of Advanced Robotic Systems* 17.2 (Mar. 2020). Publisher: SAGE Publications, p. 1729881420904303. ISSN: 1729-8814. DOI: 10.1177/1729881420904303. URL: https://doi.org/10.1177/1729881420904303 (visited on 05/18/2021).

[4] Daniil Osokin. "Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose". In: *arXiv:1811.12004 [cs]* (Nov. 2018). arXiv: 1811.12004. URL: http://arxiv.org/abs/1811.12004 (visited on 05/18/2021).

[5] Claire Lifan Chen, Ata Mahjoubfar, Li-Chia Tai, et al. "Deep Learning in Label-free Cell Classification". en. In: *Scientific Reports* 6.1 (Mar. 2016). Number: 1 Publisher: Nature Publishing Group, p. 21471. ISSN: 2045-2322. DOI: 10.1038/srep21471. URL: https://www.nature.com/articles/srep21471 (visited on 05/18/2021).

[6] Yann Lecun. "Gradient-Based Learning Applied to Document Recognition". en. In: *PROCEEDINGS OF THE IEEE* 86.11 (1998), p. 47.

[7] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". en. In: (), p. 60.

[8] Li Fei-Fei, Rob Fergus, and Pietro Perona. "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories." en. In: (), p. 9.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919. June 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, et al. "The Pascal Visual Object Classes (VOC) Challenge". en. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-009-0275-4. URL: http://link.springer.com/10.1007/s11263-009-0275-4 (visited on 05/18/2021).

[11] Bolei Zhou, Hang Zhao, Xavier Puig, et al. "Scene Parsing through ADE20K Dataset". en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 5122–5130. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.544. URL: http://ieeexplore.ieee.org/document/8100027/ (visited on 05/18/2021).

[12] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, et al. "The Role of Context for Object Detection and Semantic Segmentation in the Wild". en. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, June 2014, pp. 891–898. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.119. URL: https://ieeexplore.ieee.org/document/6909514 (visited on 05/18/2021).

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". en. In: *arXiv:1604.01685 [cs]* (Apr. 2016). arXiv: 1604.01685. URL: http://arxiv.org/abs/1604.01685 (visited on 04/09/2021).

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. "Microsoft COCO: Common Objects in Context". In: *arXiv:1405.0312 [cs]* (Feb. 2015). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312 (visited on 04/05/2021).

[15] Kevin Crowston. "Amazon Mechanical Turk: A Research Tool for Organizations and Information Systems Scholars". en. In: *Shaping the Future of ICT Research. Methods and Approaches*. Ed. by Anol Bhattacherjee and Brian Fitzgerald. IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer, 2012, pp. 210–221. ISBN: 978-3-642-35142-6. DOI: 10.1007/978-3-642-35142-6_14.

[16] Wei Guo, Bangyou Zheng, Andries B. Potgieter, et al. "Aerial Imagery Analysis Quantifying Appearance and Number of Sorghum Heads for Applications in Breeding and Agronomy". English. In: *Frontiers in Plant Science* 9 (2018). Publisher: Frontiers. ISSN: 1664-462X. DOI: 10.3389/fpls.2018.01544. URL: https://www.frontiersin.org/articles/10.3389/fpls.2018.01544/full (visited on 05/18/2021).

[17] *Detectron2 tutorial*. en. URL: https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7m5 (visited on 05/18/2021).

[18] Benjamin Sapp, Ashutosh Saxena, and Andrew Y Ng. "A Fast Data Collection and Augmentation Procedure for Object Recognition". en. In: (), p. 7.

[19] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, et al. "Simultaneous Detection and Segmentation". In: *arXiv:1407.1808 [cs]* (July 2014). arXiv: 1407.1808. URL: http://arxiv.org/abs/1407.1808 (visited on 04/08/2021).

[20] Julien Champ, Adan MoraFallas, Hervé Goëau, et al. "Instance segmentation for the fine detection of crop and weed plants by precision agricultural robots". en. In: *Applications in Plant Sciences* 8.7 (2020). _eprint: https://bsapubs.onlinelibrary.wiley.com/doi/pdf/10.1002/aps3.11373, e11373. ISSN: 2168-0450. DOI: https://doi.org/10.1002/aps3.11373. URL: https://bsapubs.onlinelibrary.wiley.com/doi/abs/10.1002/aps3.11373 (visited on 05/17/2021).

[21] Eslam Mohamed, Mahmoud Ewaisha, Mennatullah Siam, et al. "Monocular Instance Motion Segmentation for Autonomous Driving: KITTI InstanceMotSeg Dataset and Multi-task Baseline". en. In: *arXiv:2008.07008 [cs]* (Feb. 2021). arXiv: 2008.07008. URL: http://arxiv.org/abs/2008.07008 (visited on 04/09/2021).

[22] Kentaro Wada, Kei Okada, and Masayuki Inaba. "Joint Learning of Instance and Semantic Segmentation for Robotic Pick-and-Place with Heavy Occlusions in Clutter". en. In: *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 9558–9564. ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8793783. URL: https://ieeexplore.ieee.org/document/8793783/ (visited on 04/09/2021).

[23] Agrim Gupta, Piotr Dollar, and Ross Girshick. "LVIS: A Dataset for Large Vocabulary Instance Segmentation". en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 5351–5359. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00550. URL: https://ieeexplore.ieee.org/document/8954457/ (visited on 04/16/2021).

[24] Markus Suchi, Timothy Patten, David Fischinger, et al. "EasyLabel: A Semi-Automatic Pixel-wise Object Annotation Tool for Creating Robotic RGB-D Datasets". In: *arXiv:1902.01626 [cs]* (Mar. 2019). arXiv: 1902.01626 version: 2. URL: http://arxiv.org/abs/1902.01626 (visited on 04/16/2021).

[25] K. He, G. Gkioxari, P. Dollár, et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. ISSN: 2380-7504. Oct. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". en. In: *arXiv:1506.01497 [cs]* (Jan. 2016). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497 (visited on 05/18/2021).

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. "Deep Residual Learning for Image Recognition". en. In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 05/18/2021).

[28] Kaiming He, Ross Girshick, and Piotr Dollar. "Rethinking ImageNet Pre-Training". en. In: (), p. 10.

[29] Yuxin Wu, Alexander Kirillov, Francisco Massa, et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[30] Yiming Liu, Shaohua Zhang, Xiaohui Xiao, et al. "A Robotized Data Collection Approach for Convolutional Neural Networks". en. In: *Intelligent Robotics and Applications*. Ed. by YongAn Huang, Hao Wu, Honghai Liu, et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 472–483. ISBN: 978-3-319-65298-6. DOI: 10.1007/978-3-319-65298-6_43.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". en. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 04/17/2021).

[32] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. "Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection". In: *arXiv:1708.01642 [cs]* (Aug. 2017). arXiv: 1708.01642. URL: http://arxiv.org/abs/1708.01642 (visited on 04/17/2021).

[33] Arjun Singh, James Sha, Karthik S. Narayan, et al. "BigBIRD: A large-scale 3D database of object instances". en. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 509–516. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6906903. URL: http://ieeexplore.ieee.org/document/6906903/ (visited on 04/17/2021).

[34] Kevin Lai, Liefeng Bo, Xiaofeng Ren, et al. "A Large-Scale Hierarchical Multi-View RGB-D Object Dataset". en. In: (), p. 8.

[35] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, et al. "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation". In: *arXiv:2012.07177 [cs]* (Dec. 2020). arXiv: 2012.07177 version: 1. URL: http://arxiv.org/abs/2012.07177 (visited on 04/17/2021).

[36] Yosuke Toda, Fumio Okura, Jun Ito, et al. "Training instance segmentation neural network with synthetic datasets for crop seed phenotyping". In: *Communications Biology* 3 (Apr. 2020), p. 173. DOI: 10.1038/s42003-020-0905-5.

[37] Stephan R. Richter, Vibhav Vineet, Stefan Roth, et al. "Playing for Data: Ground Truth from Computer Games". In: *arXiv:1608.02192 [cs]* (Aug. 2016). arXiv: 1608.02192. URL: http://arxiv.org/abs/1608.02192 (visited on 04/17/2021).

[38] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, et al. "BlenderProc". In: *arXiv:1911.01911 [cs]* (Oct. 2019). arXiv: 1911.01911. URL: http://arxiv.org/abs/1911.01911 (visited on 04/17/2021).

[39] Shuran Song, Fisher Yu, Andy Zeng, et al. “Semantic Scene Completion from a Single Depth Image”. In: *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[40] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[41] Xuebin Qin, Zichen Zhang, Chenyang Huang, et al. “U2-Net: Going deeper with nested U-structure for salient object detection”. en. In: *Pattern Recognition* 106 (Oct. 2020), p. 107404. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107404. URL: https://www.sciencedirect.com/science/article/pii/S0031320320302077 (visited on 02/19/2021).

[42] Xuebin Qin, Zichen Zhang, Chenyang Huang, et al. “BASNet: Boundary-Aware Salient Object Detection”. en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 7471–7481. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00766. URL: https://ieeexplore.ieee.org/document/8953756/ (visited on 04/20/2021).

[43] Patrick Wspanialy. *pycococreator*. May 2018. DOI: 10.5281/zenodo.4627206. URL: https://zenodo.org/record/4627206 (visited on 04/20/2021).

[44] Adam Paszke, Sam Gross, Soumith Chintala, et al. “Automatic differentiation in PyTorch”. en. In: (Oct. 2017). URL: https://openreview.net/forum?id=BJJsrmfCZ (visited on 04/20/2021).

[45] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, et al. “CascadePSP: Toward Class-Agnostic and Very High-Resolution Segmentation via Global and Local Refinement”. In: *arXiv:2005.02551 [cs]* (May 2020). arXiv: 2005.02551. URL: http://arxiv.org/abs/2005.02551 (visited on 03/22/2021).

[46] Ekaba Bisong. “Google Colaboratory”. In: Sept. 2019, pp. 59–64. ISBN: 978-1-4842-4469-2. DOI: 10.1007/978-1-4842-4470-8_7.

[47] Shashank Tripathi, Siddhartha Chandra, Amit Agrawal, et al. “Learning to Generate Synthetic Data via Compositing”. In: *arXiv:1904.05475 [cs]* (July 2019). arXiv: 1904.05475. URL: http://arxiv.org/abs/1904.05475 (visited on 04/17/2021).

[48] Abhishek Dutta and Andrew Zisserman. “The VIA Annotation Software for Images, Audio and Video”. en. In: *Proceedings of the 27th ACM International Conference on Multimedia*. Nice France: ACM, Oct. 2019, pp. 2276–2279. ISBN: 978-1-4503-6889-6. DOI: 10.1145/3343031.3350535. URL: https://dl.acm.org/doi/10.1145/3343031.3350535 (visited on 04/23/2021).

[49] Xu Zheng, Tejo Chalasani, Koustav Ghosal, et al. “STaDA: Style Transfer as Data Augmentation:” en. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2019, pp. 107–114. ISBN: 978-989-758-354-4. DOI: 10.5220/0007353401070114. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007353401070114 (visited on 04/17/2021).

[50] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, et al. “InstaBoost: Boosting Instance Segmentation via Probability Map Guided Copy-Pasting”. In: *arXiv:1908.07801 [cs]* (Aug. 2019). arXiv: 1908.07801 version: 1. URL: http://arxiv.org/abs/1908.07801 (visited on 04/17/2021).

[51] Lingzhi Zhang, Tarmily Wen, Jie Min, et al. “Learning Object Placement by Inpainting for Compositional Data Augmentation”. en. In: *Computer Vision ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, et al. Vol. 12358. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 566–581. ISBN: 978-3-030-58600-3 978-3-030-58601-0. DOI: 10.1007/978-3-030-58601-0_34. URL: http://link.springer.com/10.1007/978-3-030-58601-0_34 (visited on 04/17/2021).

[52] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

A FIRST APPENDIX

A.1 Foreground extraction with Chroma Key

To use the Chroma Key method to extract the foreground object, the images should be clicked in front of a backdrop that does not contain any colours of the foreground object. The image is converted into Hue-Saturation-Value (HSV) format. Images in RGB format produce incorrect segmentation masks of the foreground object. For extracting the foreground the hue of the backdrop should be calculated first. This is done by taking the median hue at the edges of the image. The region where the median is calculated is 5 per cent of the image width on both side, see Figure A.2. The backdrop could have a hue value within some range. To remove the backdrop, an estimate is taken for this range as Median Hue $\pm$ 15. Unlike, RGB channels which range from 0 to 255, the Hue channel for HSV format ranges from 0 to 179 in OpenCV.

Figure A.1: Median hue is calculated from the region covered by blue rectangles on the image.

Because of the backdrop used, the foreground objects might have similar colour due to reflections from the background. As a result, the foreground objects are incorrectly segmented. Shadows and uneven lighting conditions can also lead to similar issues where the background appears in the final image, as shown in Figure. A.2 .

There are various artefacts from the foreground extraction. Along the edges of the extracted foreground object in Figure. A.2, it can be observed that the colour of the background is still present. Some parts inside the foreground object have also been eroded. Also, there are patches where the backdrop has not been removed properly. The Instance segmentation model might over fit these artefacts and would not learn to properly detect and segment objects on the real images. The artefacts can be removed using morphological transformations from OpenCV [52].

Figure A.2: An example where the foreground object is not clearly segmented