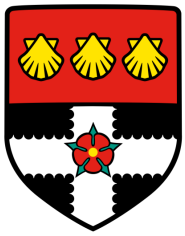

University of Reading

Department of Computer Science

Transfer learning for Instance segmentation of Waste bottles using Deep Learning Mask R-CNN

Punitha Jaikumar

Supervisor: Dr Varun Ojha

Co-Supervisor: Dr Rémy Vandaele

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Master of Science in *Computer Science*

September 14, 2020

Declaration

I, Punitha Jaikumar, of the Department of Computer Science, University of Reading, confirm that all the sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Punitha Jaikumar
September 14, 2020

Abstract

Plastic bottles constitute one of the major pollutants posing a serious threat to the environment both in oceans and land surfaces. Studies have shown that plastic bottles ranked in the top 5 pollutants, with- about 1.75 million bottles collected in 2019 [1]. The massive volume of bottles and other types of waste materials makes the automated identification and segregation of bottles a crucial task for recycling. This work studies the use of Mask RCNN, a Deep Learning Neural Network, to identify waste bottles of various shapes. The model provides detection and Instance Segmentation of object classes. The network is trained with a custom dataset comprising of bottle images downloaded from the internet and pre-processed by pixel to pixel object polygon annotation and labelled using VGG Annotator. Due to constraints both in computing power, time and limited dataset, the Transfer Learning approach is considered for this work to optimise the resource utilisation and improve model performance. This approach allows the knowledge learned by training for source task to be transferred to a target task. The study makes use of Microsoft COCO dataset trained weights on Mask R-CNN network over a dataset of 80 classes for training the custom dataset of 192 images. As part of this study, Transfer learning approach of fine-tuning with selective Data Augmentation applied to analyse model performance.

The models evaluated with MS COCO Evaluation Metrics: mean Average Precision (mAP) for Instance Segmentation and had achieved mAP[0.5:0.95:0.5] of 0.59 with the Stage 2 fine-tuned model.

Keywords: Semantic Segmentation, Object detection, Instance Segmentation

Report's total word count:14,011

[1] Ocean Conservancy Pollution Report 2019.seelink

Acknowledgements

I would like to thank my supervisor Dr Varun Ohja and Dr Rémy Vandaele for their invaluable help and support throughout this project. I would also like to thank Dr Julien Kunkel for granting access to the GPU at the University and providing timely support.

Special thanks to my family and friends for their support throughout my studies.

Publication

P. Jaikumar, R Vandaele, V Ojha. (2020) Transfer learning for Instance segmentation of Waste bottles using Deep Learning Mask R-CNN, *20th International Confernce on Hybrid Intelligent System*, (HIS'20) (in preparation).

Contents

List of Figures

List of Tables

List of Abbreviations

AP	Average Precision
CNN	Convolutional Neural Network
FP	False Positive
FN	False Negative
FCN	Fully Connected Network
FPN	Feature Pyramid Network
GPU	Graphical Processing Unit
IOU	Intersection Over Union
mAP	Mean Average Precision
MS-COCO	Microsoft Common Object in Context
NMS	Non-Max Suppression
RCNN	Region Based Convolutional Neural Network
ReLU	Rectified Linear Unit Layer
RoI	Region of Interest
RPN	Region Proposal Network
TP	True Positive
VIA	VGG Image Annotator
VOC	Visual Object Class

Chapter 1

Introduction

1.1 Background

Over the years, the plastic bottle has evolved from being the most useful material to the stage of a serious problem of pollution in land and sea (Parker, 2019). Modern world encouraged drinking beverages in a plastic bottle has led to the habit that still exists now. Plastic waste poses serious threat to environment and plastic bottles ranked in the top 5 of pollutants as per "Ocean Conservancy September" Annual report (see link).

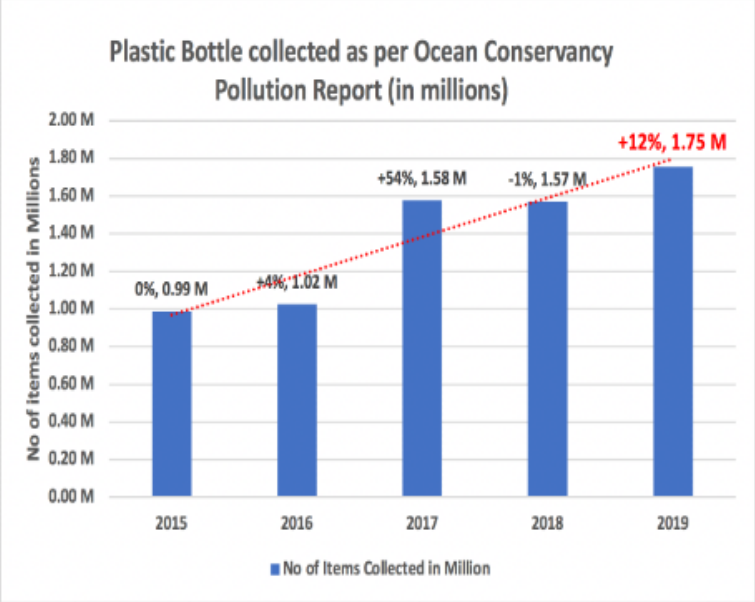

Figure 1.1: Bottle Pollution Ocean Conservancy Annual report comparison.
Ref:link

Due to population explosion and change in lifestyle, there is a surge in the use of plastic containers such as bottles resulting in the massive amount of litter polluting the environment. Many organisations around the world, both government and non-government have taken various measures to remedy the problem by applying different methods to collect and segregate the pollutants for recycling and eco-friendly disposal. Besides, the help of drones and video surveillance in urban waste management can use images to alert environmental agencies to take adequate measures. Devices such as drones and video cameras require image processing

capabilities to process images captured. Image processing algorithms have evolved over the past years, and Deep Learning algorithms have emerged rapidly. Studies have shown that Mask R-CNN can provide acceptable degree of accuracy in Instance Segmentation. (Huang et al., 2017, Garcia-Garcia et al., 2017, Zhao et al., 2019).

1.2 Problem statement

Object Segmentation in images plays a vital role in various domains. It is one of the challenging tasks in Computer Vision and requires purpose-built algorithm with the ability to detect and segment. Using the technique for plastic waste detection makes it even more challenging with limited image dataset containing damaged, broken, deformed objects. The gap in the precision of object detection compared to human vision is still large, and further work is needed to improve it to an acceptable level. With the development of the Deep Learning techniques and the success of Convolutional Neural Networks, powerful feature extraction capabilities have changed the way object detection is performed. This work is to study the performance of the algorithm in detecting the plastic waste material, bottles in particular. The range of object shapes and forms in the images used to train the algorithm is too broad to compile a comprehensive dataset. However, there are features available in deep learning frameworks to augment data to improve the performance of the model trained with a limited dataset. Transfer Learning, an approach used to re-purpose the knowledge acquired by training an algorithm for a different problem can be explored to increase the performance and optimisation of resources used.

1.3 Aims and objectives

Aims: To use transfer the learning methodology and image detection model (Mask RCNN) to perform waste plastic bottle detection.

The aim of this project is to use transfer learning methodology for Waste bottle Instance segmentation using Deep Learning Mask R-CNN. To start with creating a custom dataset by downloading Images from the internet due to the limited availability of public dataset regarding waste bottles collection for bottles segmentation. Pre-processing dataset image object annotation and labelling, image resizing. Building model pipeline with transfer learning methodology, Mask RCNN Instance Segmentation Algorithm. Evaluate the model performance using mAP (mean Average Precision) as per the COCO dataset evaluation metrics. Model Hyper-parameter tuning and testing sample image and video of waste bottles.

Objectives: This project is to use transfer learning methodology for Waste bottle Instance segmentation using Deep Learning Mask R-CNN. To create a custom dataset by downloading Images from the internet due to the limited availability of public dataset regarding waste bottles collection for bottles segmentation. Pre-processing dataset image object annotation and labelling, image resizing. Build a model pipeline with transfer learning methodology, Mask RCNN Instance Segmentation Algorithm. Evaluate the model performance using mAP (mean Average Precision) as per the COCO dataset evaluation metrics. Model Hyper-parameter tuning and testing sample image and video of waste bottles.

1.4 Proposed Work

The Proposed method Mask RCNN has achieved significant performance in many object Instance segmentation. In this work, the Transfer-learning technique of fine-tuning is applied for Waste bottle Instance segmentation on a custom dataset using the Mask RCNN algorithm using Deep Convolutional Neural Network. The algorithm applied using the TensorFlow framework and Keras implementation due to the resource constraints of dataset limited availability and computation resources Inductive transfer learning is applied in this work using pretrained coco weights. The model evaluated using COCO Evaluation method using Mean Average Precision(mAP) for object Instance segmentation performance.

Our contribution in this work would be to develop a segmentation model for bottle segmentation and analyse the model performance for the various transfer learning approaches for training. Analyse the strength of the algorithm over several dense bottle images and provide a segmentation framework and trained model.

Chapter 2

Literature Review

Object segmentation task has evolved over the years in the field of Computer vision and researched widely for its application in waste segregation for recycling purpose. Waste detection has recently gained importance with the surge in plastic pollution. Study initiated in this direction to solve the global problem of waste management with various machine learning techniques. Recent work on image-based systems have revealed the potential to find an automated solution for environment problem on both the land and water. Fulton et al. (2019) in their work concluded that the Faster RCNN performed well among the networks trained with dataset containing plastic bottle images of underwater debris. Object segmentation is a multi-step problem of detection and segmentation. Image classification involves determining the class of the objects in the images. The traditional Machine learning classification techniques of K-Nearest Neighbour(K-NN), Support Vector Machine(SVM) thresholds, Histogram based image segmentation and edge detection were part of computer vision techniques for object image segmentation.

With the recent developments, deep learning networks have outperformed traditional machine learning models. With the enhancement of Neural networks various approaches adapted to address the object classification and detection needs. Yang and Thung (2016) used the SVM and CNN for waste classification, which achieved 63% accuracy. Mittal et al. (2016) project aimed to determining whether an image is a garbage or not, and achieved a mean accuracy of 87.69% . Rad et al. (2017) achieved around 63% to 77% accuracy in waste classification on the streets using the SVM. CNN implementation on IoT for the reverse vending machine for sorting plastic waste recognition and sorting using LeNet for classification(Kokoulin et al., 2018). Classification of plastic bottles based on visual and physical features for waste management using the machine learning techniques of SVM, KNN, decision tree and Logistic regression(Kambam and Aarthi, 2019).

Wang et al. (2018) in their work, 'Bottle detection in the wild using the Low-Altitude Unmanned Aerial Vehicles' used the Faster R-CNN, SSD and YOLOv2 architecture to compare the performance and has shown that the Faster R-CNN achieved PASCAL VOC Ap precision of 90.3% followed by SSD achieving 90.1% and the YOLOv2 77.4% . The Rotational region proposed network performed better for bottles in the wild. TACO trash annotation in the context of litter detection various trash objects included as part of the public dataset creation(Proença and Simões, 2020). Hariharan et al. (2014) in their work shows that the Mask-RCNN performs well for the transparent material detection of glass and plastic.

2.1 Introduction to Machine Learning and Deep Learning

Machine Learning (ML), a subset of Artificial Intelligence is an art of programming computers to learn from data. ML algorithms help humans to learn from patterns of data to get insights about complex problems from a large amount of data. ML is broadly categorized as supervised, semi-supervised, unsupervised, and reinforcement learning. In supervised learning, the training data input to the algorithm also contains labels of the output values. Some of the important learning algorithms in supervised learning, such as Support Vector Machines, K-Nearest Neighbours, Linear Regression, Logistic Regression, and Neural networks. In unsupervised learning, the training data is unlabelled. Example Clustering algorithm K-Means, DBSCAN. (Geron, 2017). Classical Image segmentation methods are Thresholding, K-means clustering, Histogram based image segmentation, edge detection. Latest developments in Deep Learning lead to the featured based learning in algorithms.

Deep Learning techniques is a sub-field of machine learning developed from the Artificial Neural Network(ANN)(Goodfellow et al., 2016). It is a versatile, powerful and scalable technique to tackle large highly complex Machine Learning task such as speech recognition, video recommendation, gaming such as DeepMind AlphaGo. Deep Learning is hierarchical representation of Low-level, mid-level and High-Level feature for the trainable classifiers. In this video by Yann LeCunn in his talks pertaining to Deep Learning in The Institute for Scientific Computing Research (ISCR)2015. Neural networks have paved the way for the deeper learning of features. In machine learning the multidimensional arrays of data input are referred as tensors along with the multidimensional array of parameters are adapted for the learning algorithms.

Convolutional networks use the mathematical operations called convolutions, which are linear operations. Instead of matrix multiplication in at least one of the layers of neural networks convolutions are used. The convolution layer has the kernel of smaller dimension, which slides or convolves over the image matrix producing the feature map. The convolution is the sum of the element-wise multiplication of the kernel to the image matrix. The convolution layer decreases the output feature map size. The kernel stride needs to be defined for the convolution. It is the number of pixels by which the kernel shifts at a time.(Geron, 2017)

The convolutional neural network hyper-parameters being the filter size dimensions, strides-number of pixels while convolving the filter on the image and padding zeros around the image so that the edges details are learned by the neural networks.Several activation function of the convolutional neural network with smaller filter allows to learn complex non-linear transformations. The convolution layers known as receptive field activation depends upon the window in the previous layer (Geron, 2017). The process of Pooling is to learn the high-level features as it learns deeper in the network. The output of each unit in a layer is set to the maximum value of a fixed number of neighbouring units. This approach of Max pooling being introduced by Zhou and Chellappa in 1988, in IEEE International conference of neural networks. By shifting a small number of pixels the max pooling can induce a translational invariance by reducing the impact of shifting.(Goodfellow et al., 2016)

Deep Learning for Object Detection

Deep Learning in the field of computer vision developed from Image classification, Object localization, object segmentation to instance segmentation (Brownlee, 2019). Zhao et al. (2019)

in his paper on object segmentation discusses approaches adopted in the object segmentation and compares performance of various algorithms and its benchmark. Instance segmentation task of Computer vision aims to locate an object at pixel-level accuracy. Over the years the deep learning community had shown massive progress in the Object detection and the segmenting task using the features extracted from the deep learning models. AlexNet, VGGNet, ResNet, and DenseNet (Parthasarathy, 2017) gained insight into the training process to show how the network depth increased power of Representation and improved accuracy. Modern network architectures such as Inception, Resnet, and DenseNet have deeper layers but fewer features. They can perform better than their older cousins by modifying/eliminating the fully connected network architecture. Increase in the High-performance computing infrastructure lead to the rapid evolution of the object detection techniques using deep architectures. Deep learning networks used as a backbone for many state-of-art detectors to extract features for classification and localisation tasks. These detectors use benchmark datasets such as ImageNet(Deng et al., 2009), PASCAL VOC (Everingham and Winn, 2010), MS COCO (Lin et al., 2014). Network architecture such as Faster R-CNN, R-FCN, SSD, YOLO mentioned as good in consumer products such as Google photos, Pinterest visual search. Many latest algorithms optimised for accuracy by using ensemble and multi-crop methods which are often too slow for practical usage(He et al., 2016). Faster R-CNN can speed up without compromising on the accuracy and making it competitive compared to SSD(Liu et al., 2016).

2.2 Object Detection

Two Stage Detectors

The Region proposed methods of object detection are from the R-CNN family. The models pipeline detects objects in two stages: (1) generating proposal of region of interest and (2) classification objects in the proposed regions. The Region Proposal algorithms have evolved in the following order: RCNN, Fast RCNN, Faster RCNN and Mask RCNN over the years (Zhao et al., 2019). R-CNN : Region Proposal- bounding box, feature extractor for each Region proposed and classifier – linear SVM classifier model. Slow object detection with multiple pipelines leading to expensive training. Fast R-CNN developed to address the speed issues of R-CNN in 2015. Spatial pyramid pooling in Deep CNN added using the forward pass caching algorithm. VGG-16 pre-trained CNN used as feature extractor and further custom layers of Region of Pooling layer or RoI layer added to extract inputs specific to the Region (Girshick, 2015). Faster R-CNN: It is a single unified model with the Region Proposed network and Fast RCNN(Ren et al., 2015). Simultaneous Detection and segmentation initially addressed by (Hariharan et al., 2014). In their work the R-CNN architecture was used to obtain the mask of every instance to a bounding box. This enhancement led to further developments in the Object segmentation task. After this step the non-maximum suppression to the proposals applied and kept 'n' proposal per instance. Later the same authors published another paper with the Hyper column for object segmentation and fine-grained localisation. (Hariharan et al., 2015)

He et al. (2017), introduced the Mask RCNN algorithm as an extension of the Faster RCNN (Ren et al., 2015), using the two stage of Feature Pyramid Network(FPN) and the Region Proposal Network (RPN) for the pixel-based segmentation. In this task, the seminal architecture of ResNet 50 and 101 architecture which addresses the vanishing gradient problem of deeper networks based on residual blocks.(He et al., 2016), Skip connection or shortcut allow activation with 2 or 3 hops. Spatial layout of input object encoded by mask and predicted

using the pixel to pixel convolutions using FCN instead of the fully connected layers will flatten into vectors lacking spatial dimensions. Mask R-CNN has adopted the architecture of faster R-CNN with the additional features RoIAlign instead of RoIPooling and a new pipeline to mask the detected object(He et al., 2017). This technique had gained popularity as it achieves very good accuracy in object detection and classification.

Single Stage Detectors

The single-stage detectors take an approach of a simple regression problem by learning the class probabilities and bounding box coordinates and classification for the input image. (Soviany and Ionescu, 2018). YOLO family(Redmon et al., 2016), and SSD(Liu et al., 2016) belongs to this category of detector and are designed for speed and real-time use while compromising on accuracy.

2.3 Transfer Learning

Transfer Learning is a machine learning method where a learned model is used as a starting point for a similar or a new task. Training Deep learning models is an expensive process in terms of computing resource and time. Mask RCNN ResNet-50-FPN on coco trainval135k dataset required 8 GPU with 44 hours of training (Proença and Simões, 2020, Weng, 2017).Deep Learning uses this technique in Computer vision related problems to overcome the vast resource requirement including data for training network models to predict with an acceptable level of accuracy. Inductive transfer learning method is used in the Deep Learning neural network. This aspect of learning works well for general features suitable for both base and target networks(Brownlee, 2017, Torrey and Shavlik, 2010).

Pre-trained weights from already trained network can be used as the starting point. Whole or part of the model can be used for the custom training purpose. Optionally the model can be refined or adapted for the input-output pair of new tasks of interest. Instead of learning a new CNN with random initialization of weights, transfer learning used to learn from existing CNN. The final output of CNN as a feature extractor, encoding of input image vector used as an input known as CNN code, which replaces the fully connected layers. The weights changed as per the new requirements. The weights in the initial layers being low-level features will be useful in many classification tasks. While deeper layers learn the high-level features, this can be altered and retrained according to the problem definition—the weights used as a starting point of the pre-trained network for the task at hand.

The idea of fine tuning a model from an already trained on a dataset of the target dataset is mentioned in the supervised learning approach by Brownlee (2017). Instead of randomly initializing weights pre-trained model on MS COCO Dataset it is used for training of the proposed model. Transfer Learning is an optimization technique to save time and improve performance. The downside of using the technique in general is that the benefit will not be known in the domain until model developed and evaluated. A technique to use learnings of models trained on one task re-purposed for a similar task. Limited availability of training dataset and knowledge utilization from learned task to achieve good accuracy. In addition to the transfer learning the potential of Data Augmentation. It is the process of generating new input data by representing the same image object in different Representations by rotating, transforming, or mirror for training (Shorten, 2019, Sumit, 2019).

2.4 Summary

Review of related work published in the recent past on the use of object segmentation models has shed lights on different types of models analysed for various problems. Mainly the two types of models - the region proposed models performing detection in two stages and single stage models such as YOLO, SSD taking a regression/classification approach. As high accuracy is the key criteria for selection of model, Mask RCNN is chosen for this work.

Chapter 3

Mask R-CNN

This chapter explains the Mask RCNN architecture for the instance segmentation of bottles. The model implementation is based on the Mask RCNN paper [17] with fine tuning for the custom dataset. The algorithm follows the Mask R-CNN Network architecture as shown in figure 3.1.

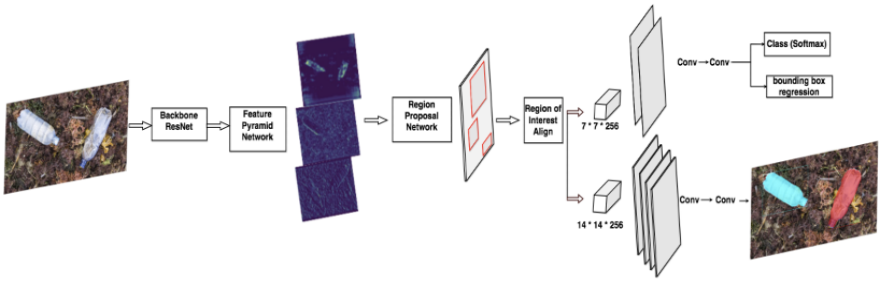

Figure 3.1: Mask R-CNN Network Architecture

3.1 Backbone Network Architecture : ResNet

The backbone network for the Instance segmentation model uses the ResNet convolutional and the Feature Pyramid Network for the initial feature extraction applying the transfer learning approach. The ResNet-convolutional neural network backbone architecture uses 1*1 Convolution used for dimensionality reduction to apply non-linearity after convolutions. It is a low dimension embedding that maps the input pixel with output channels that can be squeezed to a desired depth (Lin et al., 2013, He et al., 2016).

Batch Normalization

Normalization steps fixes the mean and the variances of the layer input to accelerate the training of deep neural network. It reduces the gradient dependence on the scale of parameters or initial values, thus reducing divergence. This process of batch normalisation regularises the model resulting in reduction of dropout(Ioffe and Szegedy, 2015). The normalization is applied as shown in the equations below

$$Mean = \mu_\beta = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{3.1}$$

$$Variance \; = \; {\sigma^2}_\beta = \; \frac{1}{m} \; \sum_{i=1}^{m} (x_i - \mu_\beta)^2 \tag{3.2}$$

$$\hat{x}_i \; = \; \frac{x_i \; - \; \mu_\beta}{\sqrt{\sigma_\beta^2 \; + \; \epsilon}} \tag{3.3}$$

$$y_i = \gamma \hat{x}_i + \beta = BN\gamma, \beta \, (x_i) \;\; where \; \gamma \; and \; \beta \; are \; learnable \; parameter. \tag{3.4}$$

ReLU Activation Function

In this work as it is a multiclass problem, Rectified Linear Units (ReLU) activation that are linear in the positive dimension and the zero in the negative dimension to generalize to avoid non-saturation of gradientsLin et al. (2017).

$$f(x) = max(0, x) \tag{3.5}$$

Residual Block

The skip connection blocks that learns functions with reference to the layer inputs. The stacked nonlinear layer fits another mapping as an addition residual block refer Figure 3.2. The Bottleneck Residual Block utilises the 1*1 convolution to create the bottleneck to make residual block thin to increase depth and reduce the number of parameters(He et al., 2016). The minibatch set as 1 because memory bottleneck while training deeper layers refer Figure 3.3 below. SoftMax output function is used as it is a multiclass classification problem of Instance segmentation. The vector of conditional probabilities generated based on the logits-previous layers output for a loss function.

Residual-Skip Connections Blocks

Residual-Skip Connections are easier to optimize than the unreferenced mapping. If the identity mapping is optimal then the residual layer is set to '0'. Skip connections allow layers to skip and connect later in the network for gradients to flow information through the network

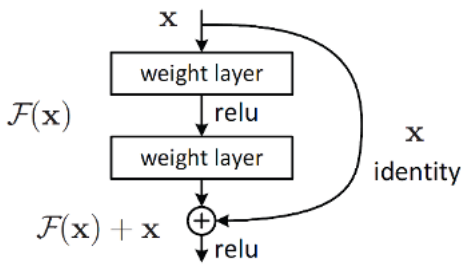

Figure 3.2: ResNet Residual Blocks
Image Ref:He et al. (2016)

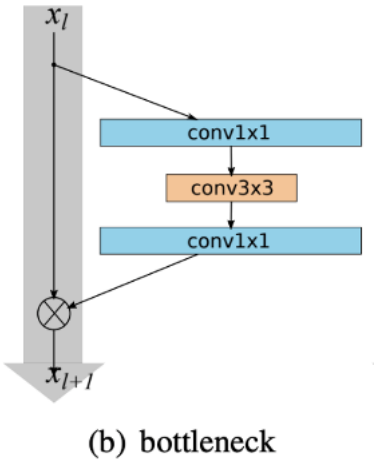

Figure 3.3: ResNet Layer Bottleneck
Image Ref:He et al. (2016)

3.2 Feature Pyramid Network (FPN)

Extracts features from Resnet backbone in bottom-up pathway and creates feature maps at top-down pathway from the ResNet backbone. The bottom-up pathway pyramid is for the backbone convolutional network ResNet each stage of convolution. Each layer output features from the previous layer will be used for the top-down pathway by lateral connection. The higher resolution feature maps are semantically stronger and is up sampled spatially coarser by using the previous layer. Same spatial size feature maps are merged from the bottom-up pathway and top-down pathway. 1 x 1 convolutions applied on the bottom-up layers and element-wise addition done to extract the features. 3 x 3 convolutions applied for each merged map to generate the final map for the prediction task refer Figure 3.4. The convolution layers C2, C3, C4 ,C5 maps to final feature maps P2, P3, P4, P5.The pyramids s hare the classifier/regressor and this generates a fixed output of dimension of width 256 for all channels (Lin et al., 2017).

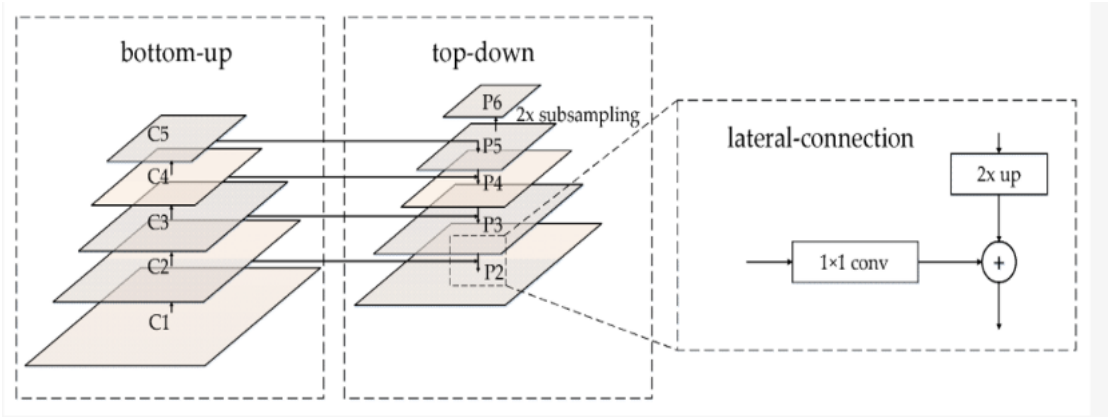

Figure 3.4: Feature Pyramid Network
Image ref:Lin et al. (2017)

3.3 Region Proposal Network (RPN)

Region Proposal Network (RPN) is Stage 1 for the Instance segmentation task in Mask RCNN. Figure 3.5 shows the RPN pipeline that runs a lightweight binary classifier on the Anchors. Anchors are boxes of different sizes generated over the image features in the FPN to extract the images that match the object, for Instance Segmentation and return probability scores of objects or no object. Anchors with a high score (positive anchors) passed to the next step for classification. The RPN network, before forwarding the positive anchors regresses for a refinement using the delta in location and size for positive anchors that does not cover objects. It shifts the bounding box and resizes to correct the boundaries of the object. The order in which the anchors are generated during training and prediction phases must be same and match those output from convolution execution. Additionally, for FPN Network the order of anchors must match the convolution layer output that predicts the anchor scores and shifts. Anchors in FPN are sorted by levels, starting from the top of the pyramid, facilitating separation by levels and within a level sorted by feature map processing sequence from left to right for each row.

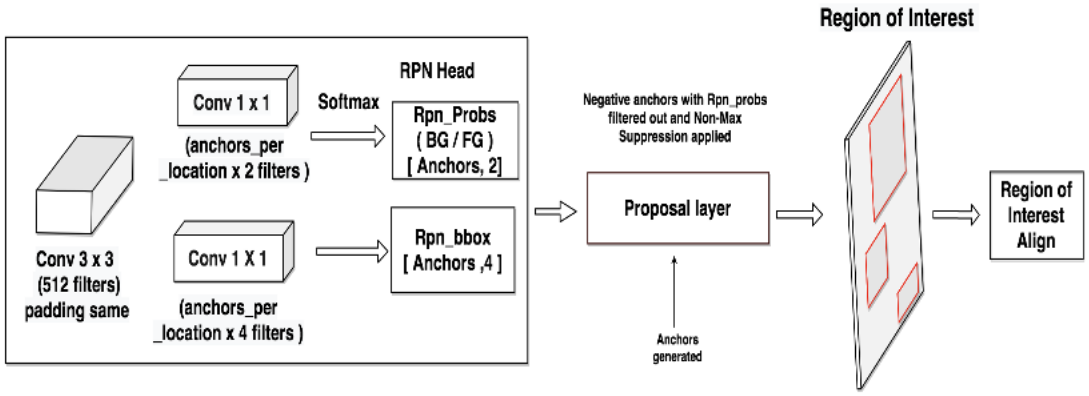

Figure 3.5: Region Proposal Network

Anchor Stride

In the FPN architecture, feature maps at the first few layers are high resolution. For input image is 1024x1024, see figure 3.6, then the feature map of the first layer is 256x256, which generates about 200K anchors (2562563). These anchors are 32x32 pixels and their stride relative to image pixels are 4 pixels, so there is lot of overlap. The anchor stride of 2 are generated for every other cell in the feature map to reduce the number of the anchors.

RPN Targets

RPN targets the training values, the initial grid of anchors generated that cover the images fully at different scales all targeted. The Intersection over Union of the anchors with the ground truth calculated for each object. The positive anchors are those with the IoU more than 05 with any ground truth object and negative anchors are those that do not cover the object more than 0.3 Intersection over Union. Neutral case that are within the range are excluded for Region Proposal network regressor calculation for image shifting and resizing to make each anchor to cover the respective ground truth object completely. The bounding box and anchor shapes are assigned to each of the object instances. Bounding box calculation is detailed in

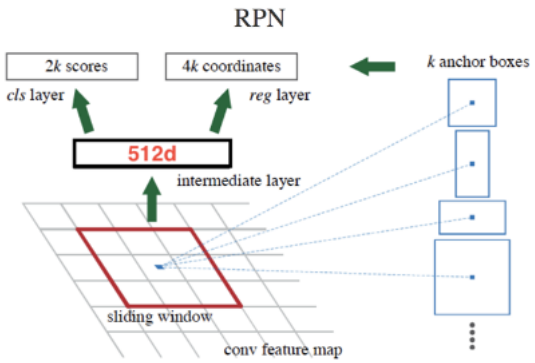

Figure 3.6: Anchor Box Generation
Image ref:Ren et al. (2015)

section below detailing the steps involved(Girshick et al., 2014). The output is passed to the proposal layer.

Bounding Boxes

The Bounding Box coordinates represented as x_1, y_1, x_2, y_2 with center point c_x, c_y $width\ w$ and $height\ h$

$$
\begin{aligned}
w &= x_2 - x_1 \\
h &= y_2 - y_1 \\
c_x &= x_1 - 0.5 * w \\
c_y &= y_1 - 0.5 * h \\
x_1 &= c_x - 0.5 * w \\
y_1 &= c_y - 0.5 * h \\
x_2 &= x_1 + w \\
y_2 &= y_1 + h
\end{aligned}
$$

Bounding Box Regression

Given the coordinate of predicted Bounding Box P $= p_x, p_y, p_w, p_h$ with center point (x , y) and width and height of predicted its corresponding ground truth box coordinates $g = (g_x, g_y, g_w, g_h)$. The regressor learns scale invariant transformation between width and height of two center and log-scale transformation.

$$
\begin{aligned}
dx &= gx - px/pw \\
dy &= (g_y - p_y)/p_h \\
d_y &= \log(g_w/p_w) \\
d_h &= \log(g_h/p_h)
\end{aligned}
$$

scale-invariant translation center P specified as $d_x,\ d_y$
width and height of P, Log-space translation set as $d_w,\ d_h$
Predicted positive bounding box P after regression is adjusted to the ground truth at inference

stage[2].

$$\hat{g}_x = p_w d_x + p_x$$
$$\hat{g}_y = p_h d_y + p_y$$
$$\hat{g}_w = p_w \exp(d_w)$$
$$\hat{g}_h = p_h \exp(d_h)$$

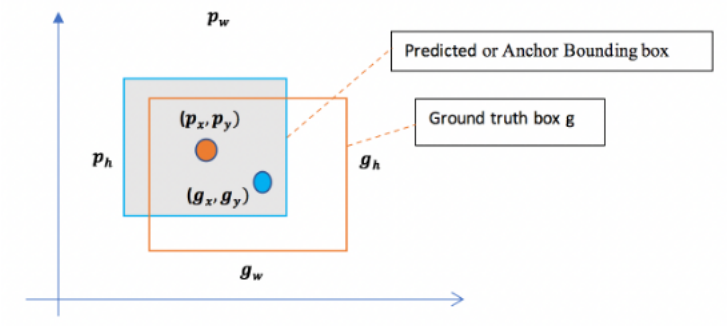

Figure 3.7: Bounding Box Regression

Bounding box Regression: Fixed position anchors generated to detect the boundaries of objects. Anchor classified to detect Positive targets. Positive anchors adjusted with Bounding box regression.

Proposal Layer

The RPN classifier output is the small feature maps which is the pooled Region of Interest of various aspect ratio of the objects in the image. This Region of pool feature map is further extracted to generate uniform RoIAlign feature maps. This stage takes the region proposals from the RPN and classifies them. Run the classifier heads on proposals to generate class probabilities and bounding box regressions. In this the class Id detected and the zero-padding trimmed . Boxes that are classified as background removed. Low confidence detections removed. This detection confidence level altered as per the detection need. Non max suppression applied based on the class to show the final detection.

RoIAlign-ROI Feature Extractors

Region of Interest Align is the process of segmentation based on the extraction of features from the ROI. The harsh quantization of RoIPool is removed by aligning the extracted features to the input. RoI Align layer is used in the mask prediction task. RoI quantised to a discrete granularity of feature maps. This further divided into spatial bins which are further quantised. The final aggregate features obtained usually by max pooling. Quantisation leads to misalignment, thus a negative effect on predicting pixel-accurate masks(He et al., 2017)Ren et al. (2015)Parthasarathy (2017). The RoIAlign layer removes the harsh quantification of extracted input features. In this layer the features extracted are aligned with the input pixels. Floating point location values in the input is computed using the Bilinear interpolation. The bilinear interpolation is performed to resample the images and the textures of the images. Screen pixel location is mapped to the corresponding point on the texture map of the images. The attributes such as the colour and transparency of the four surrounding texels (texture element) are computed and applied to the screen pixels. The process is repeated for each

pixel forming the object being textured. Appropriate intensity values to pixels are calculated by the bilinear interpolation while scaling up an image for processing. The mask target is the intersection between the ground truth mask and the ROI, where only the positive ROI considered with IoU with the ground-truth at least 0.5. The RPN graph is run to display its predictions. Coordinates normalized to [0, 1] range after NMS.

3.4 Mask Head Network-FCN

This stage takes input of the detections (refined bounding boxes and class IDs) from the previous layer and runs the mask head to generate segmentation masks for every instance. Runs the classifier heads on each proposal to generate the class probabilities and the bounding box regression. Normalized Proposals are scaled to image coordinates. Each proposal is assigned the class Id, score and binary mask.Bounding Box Refinement by class specific shifting according to shape coordinates. Low confidence detection and background classified boxes are filtered out and Non-Max Suppression applied to the anchor proposals. This is followed to the mask head to generates segmentation mask for each instance at pixel level.

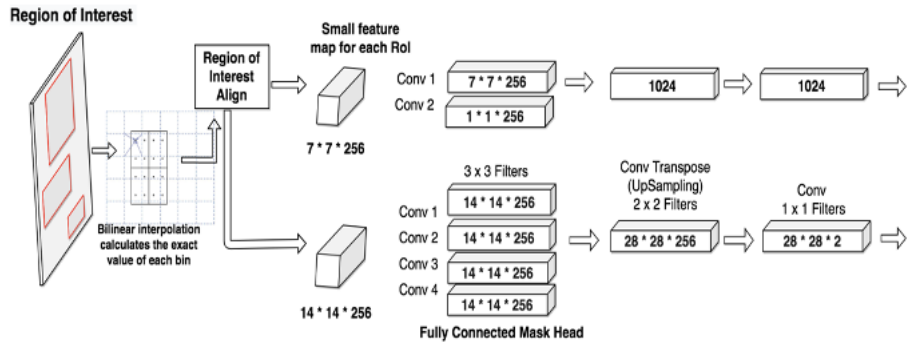

Figure 3.8: Stage2 Mask Head Network

Stage 2

High resolution images for Instance binary masks can get large while training. For example, if training with 1024 x 1024 image. Depending upon the number of instances the memory will be allocated for the mask task alone. To improve training speed, mask pixels are stored the objects in bounding box and does not store zeros around the object. Larger objects will lose a bit of accuracy.

3.5 Non-max Suppression

Region Proposal Network (RPN) for object detection, in its pipeline produces several proposals – the region of interest (ROI) with the probability score of an object in each region. These proposals are used for classification of objects. However, to help optimise classification a subset of proposals that are above a defined threshold is chosen. Non-max suppression is the most widely used algorithm to accomplish this task. Algorithm 1 shows how non-max suppression is applied. It takes a set of region proposals boxes B, set of their probability score S and a minimum threshold t for intersection over union (IOU) ratio. And returns a set of proposals, B_{max} whose IOU ratio r_{iou} with the proposal with the highest score above t.
In simple terms, the algorithm sorts the proposal boxes in the descending order of scores and adds the highest probability proposal to B_{max}. It then computes the IOU ratio r_{iou} of each of the box in the set B and adds it to result B_{max} if its above the threshold t and returns set B_{max}.

Algorithm 1 Non Max Suppression

Input: $\mathbf{B} = b_1, b_2, \ldots, b_N$
Input: $\mathbf{S} = s_1, s_2, \ldots, s_N$
Input: $\mathbf{t}$ (threshold)
Output: B_{max} (bounding boxes meeting threshold $\mathbf{t}$)

function NonMaxSuppresion($\mathbf{B}$, $\mathbf{S}$, $\mathbf{t}$)
  $B_{max} \leftarrow empty$ ▷ initialise empty array
  $B_s \leftarrow sortByScore(\mathbf{B}, \mathbf{S})$
  $b_m \leftarrow max(\mathbf{B_s})$
  B_{max} add b_m
  $N \leftarrow length(\mathbf{B_s})$
  for $i \leftarrow$1 to N **do**
    $r_{iou} \leftarrow computeIOU(\mathbf{B_i}, \mathbf{b_m})$
    if $r_{iou} \geq t$ **then** ▷ if greater or equal threshold t
      B_{max} add B_i
    end if
  end for
  return B_{max}
end function

3.6 Loss Function

The Mask R-CNN model loss is the cummulative loss of the RPN Bounding Box and class loss and the Mask R-CNN Bounding Box, Class Loss and the Mask Loss.

Loss of classification, localization and the segmentation combined in Mask RCNN

$$Loss = Lclass + LBbox + LMask \tag{3.6}$$

$$Lclass\ (P_i, P_i^*) = -P_i^* \log_{P_i} -(1 - P_i^{"})log(1 - P_i) \tag{3.7}$$

$$L(P_i, t_i) = \frac{1}{N_{class}} \sum_i L_{class}(P_i, P_i^*) + \frac{\lambda}{N_{box}} \sum_i P_i^* * L_1^{smooth}(t_i - t_i^*) \tag{3.8}$$

where L is loss;

$Lclass$ is the log loss over two classes,

$LBbox$ is Bounding Box Loss

L_1^{Smooth} is the L_1loss.

P_i Anchor object Predicted Probability i

P_i^* Ground truth label to detect objects in anchor

t_i Predicted coordinates of box

N_{class} Normalization set as 256

N_box Normalization set to the number of anchor location

λ is set to 10 to keep both the

$Lclass$ and L_Bbox equally weighted for $Lclass$ and $LBbox$

LMask added to instance segmentation as per the average of the binary cross-entropy, based on the ground truth class k the region is associated.

$$LMask = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} y_{ij} \log \hat{y}\ k_{ij} + (1 - y_{ij})\, log(1 - \hat{y}k_{ij}) \tag{3.9}$$

The predicted label of cell (i , j) is y_{ij} and the $\hat{y}kij$ is the true mask for the region of size m x m, k is predicted foreground class.
(Weng, 2017)

3.7 Evaluation metrics

Evaluation metrics for object detection and segmentation is based on the Pascal VOC metric and the MS COCO evaluation metrics. IoU (Intersection over Union)

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.10}$$

To decide how the prediction is done by the algorithm the IoU or the Jaccard Index is used. It is the intersection between the actual bounding box and the predicted bounding box divided by their union. If IoU is ¿ threshold the prediction is True Positive, otherwise False positive and False Negatives over a range of values.

Precision and Recall:

$$Recall = \frac{TruePositive\,(TP)}{TruePositive\,(TP) + FalseNegative\,(FN)} = \frac{TruePositive\,(TP)}{Ground\ Truth} \tag{3.11}$$

$$Precision = \frac{TruePositive\,(TP)}{TruePositive\,(TP) + FalsePositive\,(FP)} = \frac{TruePositive\,(TP)}{Predictions} \tag{3.12}$$

mAP of COCO Evaluation:

In COCO evaluation the IoU threshold ranges from 0.5 to 0.95 with 0.05 step size represented as AP [0.5 : 0.05 : 0.95]

AP for IoU=0.5 ad IoU=0.75 is mentioned as AP50 and AP75

$$mAPofCOCO = \frac{mAP_{0.5} \ + \ mAP_{0.55} + ... + mAP_{0.95}}{10} \tag{3.13}$$

Chapter 4

Plastic Waste Bottle Segmentation

The implementation of Plastic bottle segmentation initiated with the collection of custom dataset of bottle images for training. Mask R-CNN algorithm network pipeline implemented with Python 3, TensorFlow, Keras and OpenCV libraries. Chollet (2015).

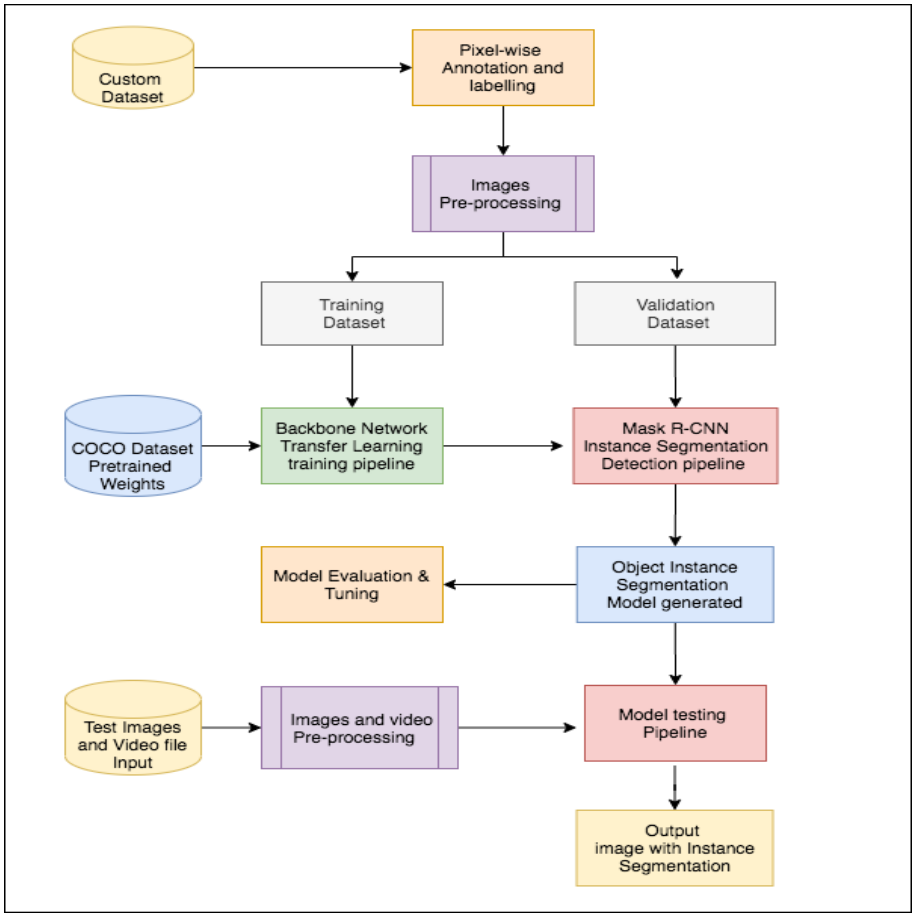

Figure 4.1: High-Level Instance Segmentation Pipeline

Figure 4.1 shows the High-Level Instance segmentation pipeline. Instance segmentation carried out using frameworks and libraries such as TensorFlow - a framework developed by Google, for high-performance calculation; Keras - the high-level open-source library for training the neural network and OpenCV - a computer vision library that facilitates image processing. The

training pipeline for transfer learning implemented by adapting the code from open-source TensorFlow implementation of Mask R-CNN.Abdulla (2017). The annotated custom dataset, along with pre-trained model weights trained on for transer learning, is feed into training pipeline for various stages of processing as discussed later in this chapter.

Pre-trained model weights trained on MS COCO dataset containing general object classes downloaded for transfer learning. Table 4.6 shows the list of key parameters configured for bottle segmentation training and inference. The Mask R-CNN algorithm feature backbone network set for the seminal architecture of ResNet 50 and 101. The strides of FPN pyramids each layer set based on the ResNet backbone set [4, 8, 16, 32, 64] for the ResNet backbome. The size of FPN fully connected layers in classification graph set to 1024 as for the bottle prediction.Top-down pyramid size set at 256 for each image.The square anchor length in pixels scale selected at (32, 64, 128, 256, 512). The image resizes dimensions set at 1024 * 1024. The detection minimum confidence set at 0.9 and the detection Non-Max Suppression threshold set at 0.3.

Learning rate set: 0.001
Learning momentum = 0.9

The number of Validation steps set at 50 after each training epoch to calculate the loss as more steps for validation will improve accuracy but can slow the training of the network. The bottle detection class set for training two class one for background and the other for foreground bottles object. The training parameters including the number of GPU set for training.

4.1 Data Pre-Processing

Segmentation of each object is a time-consuming and tedious task in this work at the initial phase. This is an important step for Instance segmentation task as it helps in defining the object pixel wise for ground truth. This will be used later by the model for Instance segmentation mask generation. Unlike other data object detectors Mask R-CNN model requires pixel wise annotation for training.

There are many publicly available tools for data annotation. Some of the popular image annotators are VIA-VGG, LabelMe, and RectLabel for Mac users. For this work the images for training are as per the COCO dataset format of pixel annotation is required. Custom dataset images are of varied sizes and in JPEG or PNG format and the pixelwise polygon graphical annotation with the VGG Annotator tool for this task. The VIA Annotator tool generates output in JSON and CSV format and saved to the disk along with the images. The ground truth mask segmented represents the spatial position and axis of each object region wise.Figure 4.2 shows each instance of the object region segmented and labeled with the annotator.

The dataset annotation also includes 2 No-instance and 2 partial annotation images. Since it is manual task and in-order imitate the human errors in annotation. Table 4.1 shows the training custom dataset Images are resized to size (1024X1024*3) to support multiple images per batch. Aspect ratio preserved for each image and padded with zero to match the one size training requirements.Images dimensions range from small image 150 * 255 to largest image with dimension 2448 * 3264 in the dataset. Pixel-wise polygon Annotation done for the image dataset for Instance segmentation training. To feed the training network both at the training

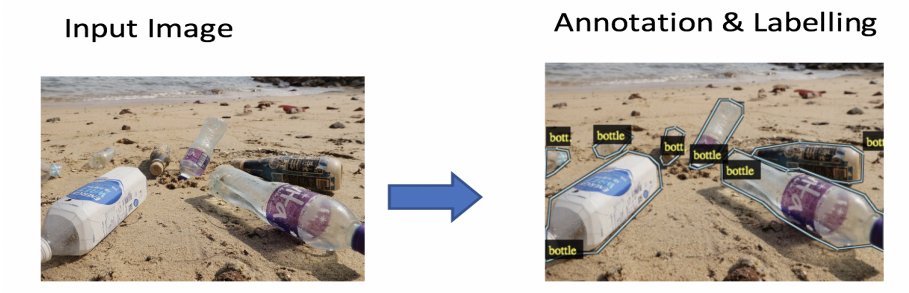

Figure 4.2: Image Annotation and Labelling

and the inference stage the Images of varied dimensions resized into square format and padded with zeros before training. All the images stored in the system in two folders with the training images and the pixel annotation and labelling information as arrays for each image in the JSON file. These image files are read one by one from the system drive for resizing.

4.2 Transfer Learning

Transfer Learning - Custom dataset training initialized with pre-trained model of MS-COCO dataset with 80 general classes. The training conducted in two phases: Phase 1 and 2 to address the complexity in implementation of Mask R-CNN algorithm requirement of memory and space. As part of phase 1 of the experiment, the models trained with backbone ResNet50 and ResNet101 for the waste bottle Instance segmentation feature training with SGD and Adam optimizer. The optimal model backbone and the optimiser, opted as per the results of the training, Phase 2 experiment initiated for enhanced transfer learning for the custom dataset.Figure4.3 shows the Phase2 transfer learning approach adopted for this segmentation task.

Transfer Learning stages
Stage 1: Heads Layer Training with all other layers frozen
Stage 2: Fine tuning layers selected or all layers
Stage 3: Extended training (optional)

Phase 1

Models trained in 2 sets: Model (1-4) : 75 training image and 15 test images for 100 epochs
Model (5-6) : 170 training image and 15 test images for 1000 epochs

The Stochastic Gradient descent optimizer and the Adam optimizer analysed with the backbone framework architecture ResNet50 and ResNet101. Models trained on Head layers in the Phase 1 training and all the remaining layers kept frozen with custom dataset and the MS_COCO weight. Model training performance evaluated to check the performance against the optimizer and the backbone architecture.
Mask R-CNN is computation heavy algorithm and due to memory constraint issues and speed bottleneck as part of the optimization approach the phase 2 models trained on Tesla V100 GPU. Model architecture ResNet101 with SGD configuration provided better results opted as deeper network architecture provided better results. The models(5-8) trained with additional

waste bottle specific images. The phase 1 model optimal performance analysed based on the loss. The phase1 optimal model parameter opted for the phase 2 transfer learning model training.

Phase 2

Phase 2 The phase2 experiments set based on the optimal model parameter of phase1, Backbone Network:ResNet101 and SGD optimizer. As part of the model optimization additional images of clustered bottle images annotated and added to the custom dataset for the Phase 2 training. In order to overcome the constraint of limited dataset, selective data augmentation adopted for training in Phase 2 to analyse model behaviour on progressive training. Figure 4.3 shows the Transfer Learning fine-tuning approach of stage 2 and 3. A set of 17 models generated refer Table 4.3 shows how the models are trained progressively using the Transfer Learning Approach.

Stage 1 Head layers trained with Data Augmentation for 30 epochs initiated for ResNet101 Backbone. Stage 2 As part of model fine tuning, the Stage 2 model trained with learned model of previous stage. $4+$ layer and ALL layers trained for 30 epoch with selective Augmentation for model training. Additionally, the $4+$ layer models trained up to 100 epochs. Stage 3 In the stage 3 training, the stage2 learned model used as base. The $4+$ layer model trained for additional 20 epochs to see whether it improves performance. With respect to the ALL layers training at stage 3, the $4+$ layer trained up to 30 epochs set as base model for training up to 30 epochs initially and then up to 100 epochs.
Incrementally the stage 3 new models trained up to 150 160 epochs for $4+$ and ALL layers respectively. Each model after training evaluated with the Coco Evaluation metric for Instance segmentation and compared against the Model Loss performance.

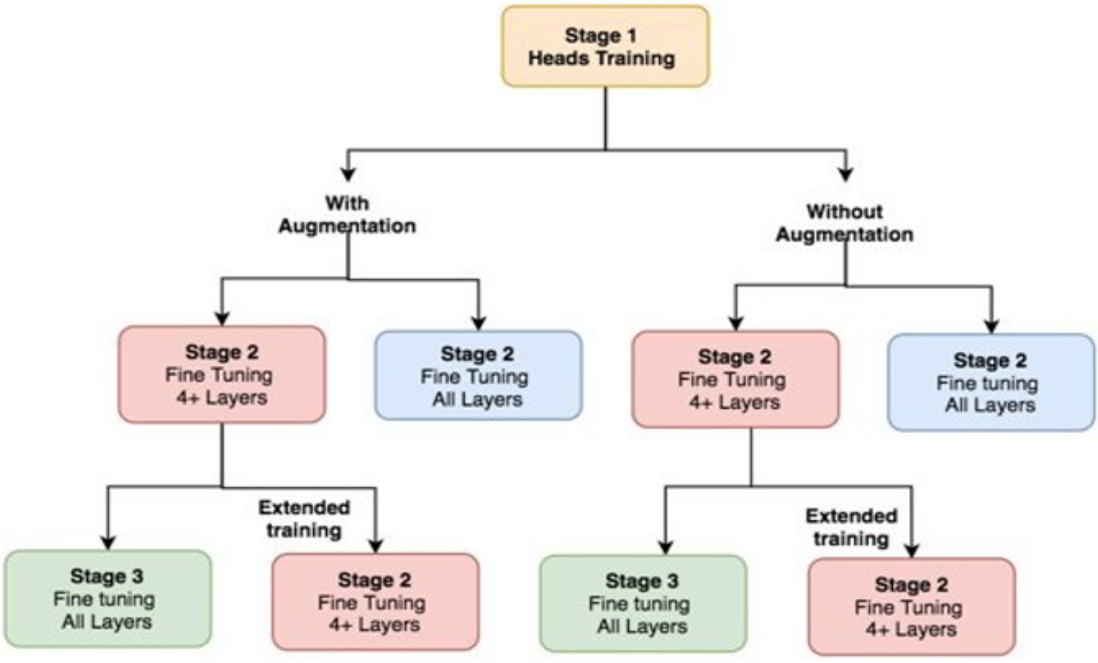

Figure 4.3: Transfer Learning Approach-Phase2 training

4.3 Model Evaluation & optimization

Phase 1 Models (1-8) performance compared on the model generalization by analysing themodel loss. Phase 2 Models (1-17) performance compared against loss metrics and all the evaluated with COCO Evaluation metric of Mean Average Precision for Instance segmentation.

4.4 Implementation Environment

Phase 1 GPU : Nvidia : GEFORCE RTX 2060 / Tesla v100
Phase 2 GPU : Nvidia Tesla v100.
TensorFlow-GPU = 1.5 setup with python 3.7 environment and OpenCV.

The initial phase 1 training done with the Nvidia GEFORCE RTX 2060 GPU for Models(1-4),Phase1 Model(5-8) Phase2 models with the high performance GPU Nvidia Tesla V100 for the experiment to handle the memory constraint and speed issues.

Dataset

The dataset consists of custom images downloaded from the internet for this work. Though there are many state-of-art data sets such as Microsoft Common Object in Common(MS-COCO) and the Pascal VOC public datasets that has general images for training and research purpose. It was noted that the availability of images pertaining to waste bottle was limited. To overcome this problem images selected and downloaded from the internet images with RGB channels collected.

The images contain single and bulk images of bottles with normal features in general and deformed features taken in various background. Table 4.1 shows the details of the breakup of the number images selected. The object detection of few bulk bottles would be a challenge for humans as well due to the occlusion and the lighting conditions, this will be a challenge for the generalization of the Object Instance segmentation for the model. All the images pixel annotated with the VGG tool to match the official coco dataset format to identify the class names to feed into the network.

Table 4.1: Custom Dataset of bottles downloaded from Internet

Type	No of Images	Waste Bottle Specif Images	Total Bottle Instance Count	Bulk Bottle Instance Count	Image Range-min pixel	Image Range-max pixel	Images Resized (RGB)
Train	177	102	712	561	33750	7990272	1024*1024*3
Val	15	3	113	104	535824	2250000	1024*1024*3

Table 4.2: Phase 1 Model(1-8) Parameters

Ref_No	Model Name	Backbone	Epoch	Optimiser	Learning Rate	Steps	Layer
1	Res101S_1	ResNet101	30	SDG	0.001	100	Heads
2	Res101A_2	ResNet101	30	Adam	0.001	100	Heads
3	Res50A_2	ResNet50	30	Adam	0.001	100	Heads
4	Res50S_1	ResNet50	30	SDG	0.001	100	Heads
5	101SGD	ResNet 101	30	SGD	0.001	1000	Heads
6	101Adam	ResNet 101	30	Adam	0.001	1000	Heads
7	50SGD	ResNet 50	30	Adam	0.001	1000	Heads
8	50Adam	ResNet 50	30	SGD	0.001	1000	Heads

Table 4.3: Phase 2 Model(1-17) Parameters

Backbone Architecture:Resnet 101, Optimizer : SGD, No of Steps :1000, Learning Rate : 0.001							
Ref_No	Model _Name	Weights	Transfer Learning Stage	Training Layers	Augment-ation	Epoch	Total Epochs
1	1SMODEL_OS	COCO	1	HEADS	Y	30	30
2	2SMODEL_OS1	1SMODEL_OS_30	2	4+	N	1	31
3	2SMODEL_OS2	1SMODEL_OS_30	2	4+	Y	1	31
4	2SMODEL_OS3	1SMODEL_OS_30	2	ALL	Y	30	60
5	2SMODEL_OS7	1SMODEL_OS_30	2	ALL	N	30	60
6	**2SMODEL_OS4**	1SMODEL_OS_30	2	4+	Y	30	60
7	2SMODEL_OS12	**2SMODEL_OS4_30**	2	4+	Y	**100**	130
8	3SMODEL_OS13	2SMODEL_OS12_100	**3**	ALL	Y	20	150
9	3SMODEL_OS14	2SMODEL_OS12_100	**3**	4+	Y	20	150
10	3SMODEL_OS8	**2SMODEL_OS4_30**	3	ALL	Y	30	90
11	3SMODEL_OS9	3SMODEL_OS8_30	**3**	ALL	Y	100	160
12	**2SMODEL_OS5**	1SMODEL_OS_30	2	4+	N	30	60
13	2SMODEL_OS6	**2SMODEL_OS5_30**	2	4+	N	**100**	130
14	3SMODEL_OS10	2SMODEL_OS6_100	**3**	ALL	N	20	150
15	3SMODEL_OS11	2SMODEL_OS6_100	**3**	4+	N	20	150
16	3SMODEL_OS15	**2SMODEL_OS5_30**	3	ALL	N	30	90
17	3SMODEL_OS16	3SMODEL_OS15_30	**3**	ALL	N	100	160

Table 4.4: ResNet Training Layers

Layer name	50-layer	101-layer	Output size	Activation
Conv2D Convolution 1 _x	7 7,64 stride 2 3× $3maxpool$ stride 2 Padding='same'	7 7,64 stride 2 3× $3maxpool$ stride 2 Padding='same'	12× 12	relu
Convolution 2_ x	$\begin{bmatrix}1\times\ 1, & 64\\ 3\times\ 3, & 64\\ 1\times\ 1 & 256\end{bmatrix}\times 3$	$\begin{bmatrix}1\times\ 1, & 64\\ 3\times\ 3, & 64\\ 1\times\ 1 & 256\end{bmatrix}\times 3$	56× 56	relu
Convolution 3_x	$\begin{bmatrix}1\times\ 1, & 128\\ 3\times\ 3, & 128\\ 1\times\ 1 & 512\end{bmatrix}\times 4$	$\begin{bmatrix}1\times\ 1, & 128\\ 3\times\ 3, & 128\\ 1\times\ 1 & 512\end{bmatrix}\times 4$	28× 28	relu
Convolution 4_x	$\begin{bmatrix}1\times\ 1, & 256\\ 3\times\ 3, & 256\\ 1\times\ 1 & 1024\end{bmatrix}\times 6$	$\begin{bmatrix}3\times\ 3, & 64\\ 1\times\ 1, & 64\\ 1\times\ 1 & 256\end{bmatrix}\times 23$	14× 14	relu
Convolution 5_x	$\begin{bmatrix}3\times 3, & 512\\ 1\times\ 1, & 512\\ 1\times\ 1 & 2048\end{bmatrix}\times 3$	$\begin{bmatrix}3\times\ 3, & 64\\ 1\times\ 1, & 64\\ 1\times\ 1 & 2048\end{bmatrix}\times 3$	7× 7	relu
Final Layer uses			1× 1	SoftMax

Table 4.5 shows the number of trainable parameters for the ResNet backbone architecture for ResNet50 and ResNet101. Table 4.4 shows the number of trainable layers for the ResNet architecture.

Table 4.5: ResNet Trainable parameters

Architecture	ResNet 50	ResNet 101
Trainable params:	44,603,678	63,621,918
Non-trainable params:	59,264	111,488
Total params:	44,662,942	63,733,406

Table 4.6: Configuration

PARAMETER	VALUES
IMAGES_PER_GPU	2
STEPS_PER_EPOCH	1000
VALIDATION_STEPS	50
NUM_CLASSES	2
BACKBONE	"resnet101" or "resnet50"
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
FPN_CLASSIF_FC_LAYERS_SIZE	1024
TOP_DOWN_PYRAMID_SIZE	256
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_STRIDE	2
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
PRE_NMS_LIMIT	6000
POST_NMS_ROIS_TRAINING	2000
POST_NMS_ROIS_INFERENCE	1000
TRAIN_ROIS_PER_IMAGE	512 # phase1 200
MAX_GT_INSTANCES	512 # phase1 150
DETECTION_MAX_INSTANCES	512
DETECTION_MIN_CONFIDENCE	0.9 # tuning options (0.7 , 0.5)
DETECTION_NMS_THRESHOLD	0.3
LEARNING_RATE	0.001
LEARNING_MOMENTUM	0.9
WEIGHT_DECAY	0.0001

Configuration

BACKBONE_STRIDES The strides of FPN based on the Resnet backbone.

FPN_CLASSIF_FC_LAYERS_SIZE Classification graph fully-connected layers size

RPN_ANCHOR_RATIO 1:value of square anchor and 0.5 and 2 for wide or narrow anchor.

PRE_NMS_LIMIT ROI kept after the top K anchor filtering before the NMS.

TRAIN_ROI_PER_IMAGE ROI to feed to the classifier/mask heads for the phase 1 experiments set as 200 Phase 2 experiments set as 512

POST_NMS_ROIS_TRAINING and POST_NMS_ROIS_INFERENCE ROI after the NMS suppression for the the training and inference stage.

MAX_GT_INSTANCES GT instances to use for one image.

DETECTION_MIN_CONFIDENCE set at 0.9 for training and validation and parameter value 0.7 and 0.5 used for tuning the model performance.

Chapter 5

Results and Discussion

Training and Validation

5.1 Phase 1 Model Results

Table 5.1 shows the Model (1-4) training and validation loss in phase 1 training. The models trained with ResNet backbone of 101 layers with the Stochastic Gradient Descent (SDG) optimizer performed better than the models trained on Adam optimizer. Model ResNet101S_1 with Ref_ No 1 has the least training loss of 0.12 compared to the other models. SGD Model trained on ResNet50 with Ref_ No 3 has the least validation loss of 0.59. Comparatively the SGD optimizer models trained on ResNet101 framework architecture better than the ResNet50.

Table 5.1: Phase 1 Model(1-8) Loss Comparison

	1	2	3	4	5	6	7	8
Model_Name	**Res101S_1**	**Res101A_2**	**Res50S_1**	**Res50A_2**	**101SGD**	**101Adam**	**50SGD**	**50Adam**
Backbone:ResNet	101	101	50	50	101	101	50	50
Epoch	30	30	30	30	30	30	30	30
Optimiser	SDG	Adam	SDG	Adam	SGD	Adam	SGD	Adam
Learning Rate	0.001	0.001	0.001	0.001	0.01	0.01	0.01	0.01
Steps	100	100	100	100	1000	1000	1000	1000
loss	0.1164	0.1344	0.1303	0.1234	0.0381	0.0482	0.0571	0.0572
rpn_bbox_loss	0.0213	0.0236	0.0269	0.0219	0.0032	0.0069	0.0116	0.0089
rpn_class_loss	0.0024	0.0031	0.0033	0.0032	0.0002	0.0003	0.0003	0.0004
mrcnn_bbox_loss	0.0145	0.0176	0.0162	0.0166	0.0015	0.0034	0.0042	0.0037
mrcnn_class_loss	0.0124	0.0174	0.0138	0.0144	0.0027	0.0049	0.0050	0.0072
mrcnn_mask_loss	0.0658	0.0725	0.0702	0.0674	0.0305	0.0328	0.0360	0.0370
val_loss	0.8746	0.7824	0.5927	0.848	1.0433	0.8682	2.3748	6.4144
val_rpn_bbox_loss	0.2228	0.2296	0.2301	0.2235	0.6329	0.5886	1.0830	2.7118
val_rpn_class_loss	0.0133	0.0176	0.0163	0.0156	0.0612	0.0008	0.7044	2.7568
val_mrcnn_bbox_loss	0.1788	0.1793	0.1143	0.1828	0.0584	0.0367	0.0969	0.1642
val_mrcnn_class_loss	0.0709	0.0520	0.0572	0.0522	0.0554	0.0176	0.2718	0.4057
val_mrcnn_mask_loss	0.3888	0.3038	0.1747	0.3739	0.2354	0.2245	0.2187	0.3760

Model 5-8 trained with 1000 steps per epoch and backbone architecture of ResNet101 and ResNet50 using SGD and Adam optimizers. Comparatively the ResNet101 model with SGD optimizer performed better with the loss value of 0.04 where are the model didn't perform

well for the validation data. The ResNet101 model with Adam optimizer achieved the least validation loss.

Overall from the Phase 1 Models (1-8), the models trained on SGD performed better with the ResNet101 Architecture. Deeper the backbone architecture the models performed better.

5.2 Phase 2 Models Result

Table 5.2 shows the results of the Fine-tuned models (1-17) training and validation loss. Overall the training loss for all models are below 0.15 however, the validation loss below 1.0 are for the trained models with Ref_ No 3,4,13,14,15,16,17.

The models 14 to 17 models are Stage2 and Stage3 models without Augmentation. This could be due to model learning fine tuning feature learning of the model. Table 5.3,5.4,5.5 shows model loss pertaining to each model.

Table 5.2: Phase 2 Model Loss Comparison

Ref_No	Model _Name	Training Layers	Augmentation	Epoch	Training Loss	Val Loss	Avg_Loss	
1	1SMODEL_OS	HEADS	Y	30	0.0963	3.1846	1.64	>1
2	2SMODEL_OS1	4+	N	1	0.1511	2.1080	1.13	>1
3	2SMODEL_OS2	4+	Y	1	0.1387	0.6376	**0.39**	<1
4	2SMODEL_OS3	ALL	Y	30	0.0696	1.2175	**0.64**	<1
5	2SMODEL_OS7	ALL	N	30	0.0482	2.4030	1.23	>1
6	2SMODEL_OS4	4+	Y	30	0.0627	3.0317	1.55	>1
7	2SMODEL_OS12	4+	Y	100	0.0295	3.0008	1.52	>1
8	3SMODEL_OS13	ALL	Y	20	0.0263	3.9649	2	>1
9	3SMODEL_OS14	4+	Y	20	0.0354	6.2123	3.12	>1
10	3SMODEL_OS8	ALL	Y	30	0.0536	4.2743	2.16	>1
11	3SMODEL_OS9	ALL	Y	100	0.0244	3.3875	1.71	>1
12	2SMODEL_OS5	4+	N	30	0.0369	2.0441	1.04	>1
13	2SMODEL_OS6	4+	N	100	0.0370	1.6917	**0.86**	<1
14	3SMODEL_OS10	ALL	N	20	0.0259	0.4858	**0.26**	<1
15	3SMODEL_OS11	4+	N	20	0.0277	1.4698	**0.75**	<1
16	3SMODEL_OS15	ALL	N	30	0.0375	0.4325	**0.23**	<1
17	3SMODEL_OS16	ALL	N	100	0.0274	1.1998	**0.61**	<1

Table 5.3: Phase2 Model(1-6) Loss

Model _Name	1SMODEL _OS	2SMODEL _OS1	2SMODEL _OS2	2SMODEL _OS3	2SMODEL _OS7	2SMODEL _OS4
Ref_No	1	2	3	4	5	6
Loss Value	0.0963	0.1511	0.1387	0.0696	0.0482	0.0626
mrcnn_bbox_loss	0.0098	0.0196	0.0143	0.0039	0.0029	0.0049
mrcnn_class_loss	0.014	0.0256	0.0178	0.0078	0.0071	0.0096
mrcnn_mask_loss	0.0498	0.0707	0.0618	0.0399	0.0337	0.0414
rpn_bbox_loss	0.0192	0.0317	0.0381	0.0177	0.0041	0.0061
rpn_class_loss	0.0036	0.0035	0.0067	0.0004	0.0003	0.0006
Val Loss	3.1846	2.108	0.6376	1.2175	2.403	3.0317
val_mrcnn_bbox_loss	0.5285	0.4472	0.1327	0.1646	0.1835	0.2717
val_mrcnn_class_loss	0.28	0.7015	0.0304	0.0948	0.6523	0.3274
val_mrcnn_mask_loss	1.7244	0.6099	0.3515	0.5934	0.3678	1.3856
val_rpn_bbox_loss	0.5766	0.3408	0.1199	0.3499	0.7483	0.8059
val_rpn_class_loss	0.0752	0.0088	0.003	0.0147	0.4511	0.2412

Table 5.4: Phase2 Model(7-12) Loss

Model _Name	2SMODEL _OS12	3SMODEL _OS13	3SMODEL _OS14	3SMODEL _OS8	3SMODEL _OS9	2SMODEL _OS5
Ref_No	7	8	9	10	11	12
Loss Value	0.0295	0.0263	0.0354	0.0536	0.0244	0.0369
mrcnn_bbox_loss	0.001	0.0008	0.0014	0.0028	0.0008	0.0022
mrcnn_class_loss	0.0037	0.0024	0.0053	0.0073	0.0027	0.0045
mrcnn_mask_loss	0.0229	0.0221	0.0253	0.0339	0.0197	0.0264
rpn_bbox_loss	0.0017	0.0008	0.0033	0.0094	0.0011	0.0034
rpn_class_loss	0.0001	0.0001	0.0001	0.0002	0.0001	0.0006
Val Loss	3.0008	3.9649	6.2123	4.2743	3.3875	2.0441
val_mrcnn_bbox_loss	0.1501	0.2419	0.1652	0.4161	0.1105	0.2944
val_mrcnn_class_loss	0.4443	0.5374	0.5327	0.6067	0.4523	0.1958
val_mrcnn_mask_loss	0.7722	1.2467	0.385	2.2337	0.5617	0.8089
val_rpn_bbox_loss	0.8742	1.0401	2.3701	0.6291	0.8476	0.5459
val_rpn_class_loss	0.7600	0.8988	2.7594	0.3886	1.4155	0.1991

Table 5.5: Phase2 Model(13-17) Loss

Model _Name	2SMODEL _OS6	3SMODEL _OS10	3SMODEL _OS11	3SMODEL _OS15	3SMODEL _OS16
Ref_No	13	14	15	16	17
Loss Value	0.0370	0.0259	0.0277	0.0375	0.0274
mrcnn_bbox_loss	0.0015	0.0009	0.0009	0.0020	0.0011
mrcnn_class_loss	0.0076	0.0035	0.0036	0.0063	0.0037
mrcnn_mask_loss	0.0254	0.0204	0.0212	0.0275	0.0207
rpn_bbox_loss	0.0023	0.0011	0.0016	0.0016	0.0019
rpn_class_loss	0.0001	0.0001	0.0003	0.0001	0.0000
Val Loss	1.6917	0.4858	1.4698	0.4325	1.1998
val_mrcnn_bbox_loss	0.0862	0.0477	0.1325	0.0406	0.0532
val_mrcnn_class_loss	0.1990	0.0516	0.2390	0.0428	0.0285
val_mrcnn_mask_loss	0.4432	0.2987	0.7190	0.2901	0.5164
val_rpn_bbox_loss	0.4148	0.0849	0.2240	0.0586	0.5944
val_rpn_class_loss	0.5484	0.0028	0.1553	0.0004	0.0072

5.3 Phase 2 Model Evaluation results

Table 5.6 shows the Phase 2 Models (1-17) mean Average Precision(mAP) results comparison. The Stage 2 model with Ref_ No 13 performed well over all the models trained with the Transfer Learning approach. It achieved AP50 of 0.74 accuracy and mAP[0.5:0.95:05] of 0.59 Accuracy in Instance Segmentation as per the MS Coco Evaluation metric. The model performance at stage 3 did not improve, this could be to the inadequate dataset. The model needs more training data to improve its precision.

Table 5.6: Phase2 Model(1-17) Evaluation Performance

Ref _No	Model_Name	AP50	AP75	AP90	Mean_AP [AP50 ,AP75, AP90]	mAP50 _Train	mAP50 _Val	Avg_mAP [Train, Val]	mAP _VAL [0.5:0.95:0.5]
1	1SMODEL_OS	0.7600	0.7338	0.1389	0.5442	*0.9696*	**0.7600**	86%	
2	2SMODEL_OS1	0.6596	0.6004	0.1522	0.4708	*0.8519*	0.6596	76%	
3	2SMODEL_OS2	0.7806	0.6942	0.3111	0.5953	*0.8953*	**0.7806**	84%	
4	2SMODEL_OS3	0.7181	0.6339	0.2491	0.5337	0.9907	0.7181	85%	0.54884
5	2SMODEL_OS7	0.7076	0.6284	0.3565	0.5642	0.993	0.7076	85%	0.55668
6	2SMODEL_OS4	0.6946	0.6339	0.2491	0.5259	0.9924	0.6946	84%	0.54535
7	2SMODEL_OS12	0.6789	0.6197	0.3713	0.5566	0.9946	0.6789	84%	0.53608
8	3SMODEL_OS13	0.7049	0.6433	0.3505	0.5662	0.9949	0.7049	85%	0.55398
9	3SMODEL_OS14	0.7044	0.64	0.3502	0.5649	0.9948	0.7044	85%	0.55003
10	3SMODEL_OS8	0.6907	0.6323	0.3034	0.5421	0.9944	0.6907	84%	0.54359
11	3SMODEL_OS9	0.691	0.6316	0.3081	0.5436	0.9929	0.691	84%	0.54292
12	2SMODEL_OS5	0.7168	0.6395	0.3136	0.5566	0.9927	0.7168	85%	0.56281
13	**2SMODEL_OS6**	0.7441	0.6461	0.4118	0.6007	0.9948	**0.7441**	**87%**	**0.59119**
14	3SMODEL_OS10	0.7107	0.6397	0.4729	0.6078	0.9948	0.7107	85%	0.56429
15	3SMODEL_OS11	0.7159	0.6649	0.4899	0.6236	0.9948	0.7159	86%	0.58240
16	3SMODEL_OS15	0.7111	0.6372	0.4052	0.5845	0.9931	0.7111	85%	0.56450
17	3SMODEL_OS16	0.7058	0.6274	0.4059	0.5797	0.9948	0.7058	85%	0.55567

Table 5.7 shows the hyper-parameter tuning of the Detection_ min_ confidence to 0.5 and 0.7 for the model 2SMODEL_ OS6. This tuning achieves a marginal improvement in the mAP_ val [0.5:0.95:0.5] of 0.35% and 0.17% over the Detection_ min_ confidence of 0.9.

Table 5.7: Model : 2SMODEL_OS6 - Fine tuning

Detection_Min_Confidence	50%	70%	90%
mAP_val	0.7542	0.7486	**0.7441**
mAP_val [0.5 : 0.5 : 0.95]: (after tuning)	**0.5947**	**0.5929**	0.5912
mAP Increase %	+0.35%	+0.17%	-

Detection Results of Phase 2 Model 2SMODEL_ OS6

The results from various stages of the Mask R-CNN detection pipeline from the RPN network to the Classification and Mask head is discussed below.

RPN generates the anchors boxes on the feature extracted by the Feature pyramid Network for the sample image. Stage1 Anchors : Count: 261888 Scales: (32, 64, 128, 256, 512) ratios: [0.5, 1, 2]
Anchors per Cell: 3
Levels: 5

Level 0.	Anchors: 196608	Feature map Shape:	[256 256]
Level 1.	Anchors: 49152	Feature map Shape:	[128 128]
Level 2.	Anchors: 12288	Feature map Shape:	[64 64]
Level 3.	Anchors: 3072	Feature map Shape:	[32 32]
Level 4.	Anchors: 768	Feature map Shape:	[16 16]

RPN

target_rpn_match	shape:	(261888,)	min:	-1	max:	1
target_rpn_ bbox	shape:	(256, 4)	min:	-2.98311	max:	3.20408
positive_anchors	shape:	(94, 4)	min:	2.74517	max:	1066.51
negative_anchors	shape:	(162, 4)	min:	-32	max:	1066.039
neutral anchors	shape:	(261632, 4)	min:	-362.039	max:	1322.039
refined_anchors	shape:	(94, 4)	min:	1	max:	1023

Figure 5.1: Anchor Positive Anchors

Multiple anchor boxes of various aspect ratio generated with the limit set to 200 for this example for each input from the feature pyramid network.Figure 5.2 shows the positive anchor after refinement based on the bottles positioning within the box.

Stage2 Proposal classification

Figure 5.2: Top Anchors with refinement clipped to image boundaries

rpn_ class	shape: (1, 261888, 2)	min: 0.00000	max: 1.00000	float32
pre_ nms_ anchors	shape: (1, 6000, 4)	min: -0.35390	max: 1.29134	float32
refined_ anchors	shape: (1, 6000, 4)	min: -0.87962	max: 2.07677	float32
refined_ anchors_ clipped	shape: (1, 6000, 4)	min: 0.00000	max: 1.00000	float32
post_ nms_ anchor_ ix	shape: (997,)	min: 0.00000	max: 5998.0	int32
proposals	shape: (1, 1000, 4)	min: 0.00000	max: 1.00000	float32

proposals shape:	(1, 1000, 4)	min: 0.00000	max: 1.00000	float32
probs shape:	(1, 1000, 2)	min: 0.00000	max: 1.00000	float32
deltas shape:	(1, 1000, 2, 4)	min: -3.68670	max: 3.52885	float32
masks shape:	(1, 100, 28, 28, 2)	min: 0.00000	max: 1.00000	float32
detections shape:	(1, 100, 6)	min: 0.00000	max: 1.00000	float32

10 detections 997 Valid proposals out of 1000
59 Positive ROIs [('BG', 941), ('bottle', 59)]
Applying Bounding Box refinement and Class specific bounding Box shift

roi_ bbox_ specific	shape: (1000, 4)	min:	-2.93513	max:	3.52885	float32
refined_ proposals	shape: (1000, 4)	min:	-52.00000	max:	1348.00000	int32

Generating Mask:
Get predictions of mask head
Get detection class IDs. Trim zero padding.
Masks

Instance binary mask generated each extracted feature maps in the RoIAlign for the masking of object instance rather than the image. Figure **??** shows the masking of

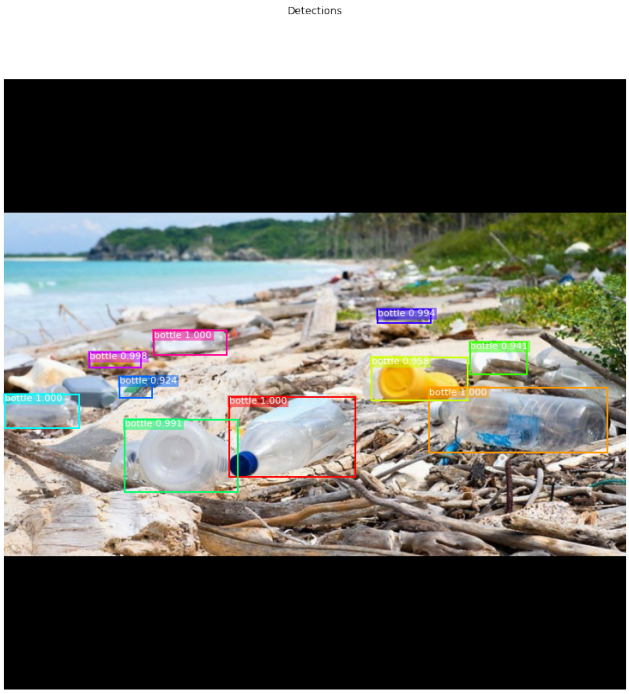

Figure 5.3: Bounding Box detected and class assigned

detections	shape: (1, 100, 6)	min:	0.00000	max:	1.00000	float32
masks	shape: (1, 100, 28, 28, 2)	min:	0.00000	max:	1.00000	float32

image	shape: (1024, 1024, 3)	min:	0.00000	max:	255.00000	uint8
molded_ images	shape: (1, 1024, 1024, 3)	min:	-123.70000	max:	151.10000	float64
image_ metas	shape: (1, 14)	min:	0.00000	max:	1024.00000	int64
anchors	shape: (1, 261888, 4)	min:	-0.35390	max:	1.29134	float32

image	shape: (1024, 1024, 3)	min:	0	max:	255	uint8
molded_images	shape: (1, 1024, 1024, 3)	min:		max:	151.1	float64
image_metas	shape: (1, 14)	min:	0	max:		int64
anchors	shape: (1, 261888, 4)	min:	-0.3539	max:	1.29134	float32
gt_class_id	shape: (19,)	min:	1	max:	1	int32
gt_bbox	shape: (19, 4)	min:	1	max:		int32
gt_mask	shape: (1024, 1024, 19)	min:	0	max:	1	bool

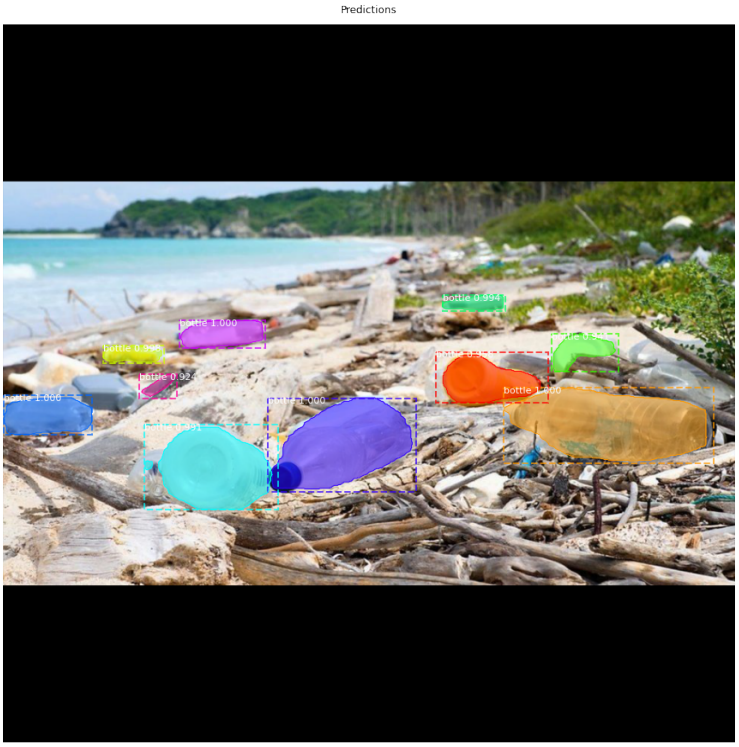

Figure 5.4: Mask segmentation

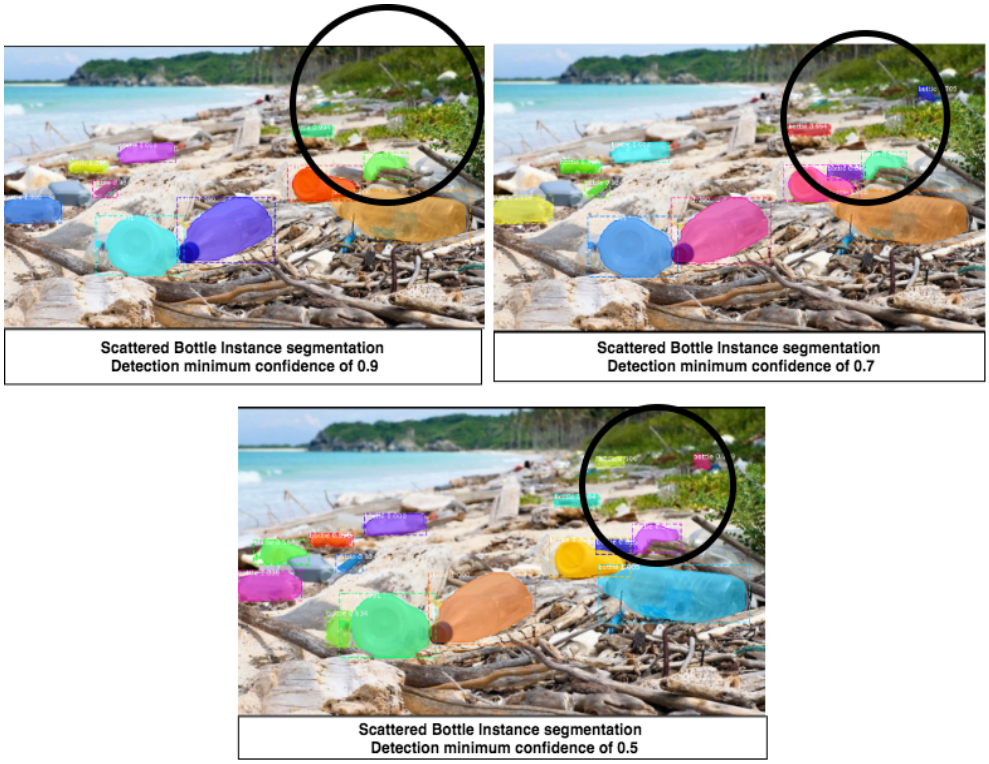

Figure 5.5: Scattered bottle segmentation

Figure 5.5, shows the scattered bottle instance segmentation at 0.9 Detection minimum confidence. However, the model segments extra bottles with good precision with low Detection minimum confidence of 0.7 and 0.5. respectively. While the scattered bottle images are detected over the threshold of 0.5 detection minimum confidence, the overall the model performs better.

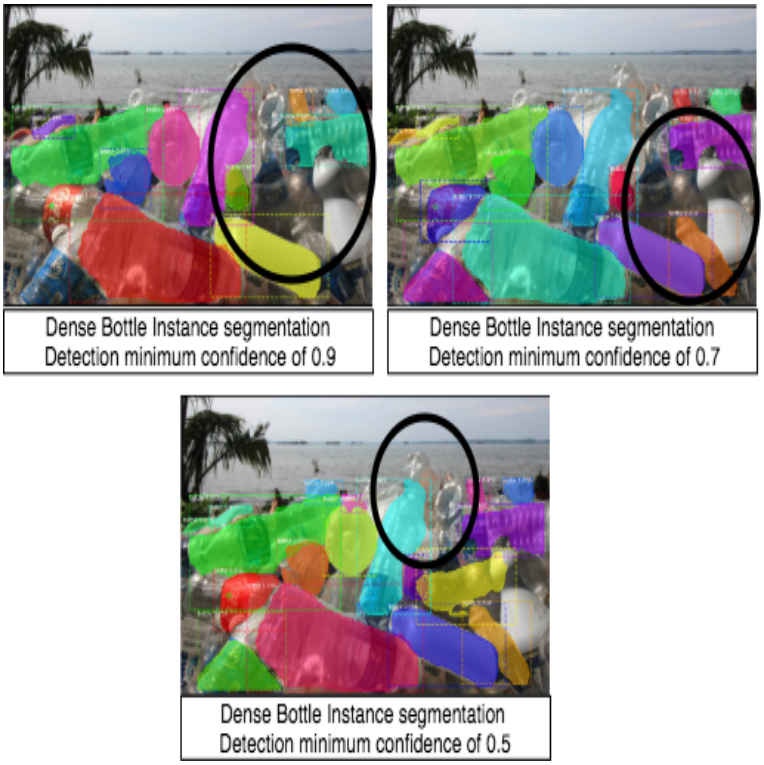

Figure 5.6: Dense bottle segmentation

Figure 5.6 shows the Instance segmentation of dense overlapping bottles. The number of instances segmented by the model increases with the decrease in the detection minimum confidence level. The image still has undetected bottle instances, highlighted with circle in the image for each output. Clearly more bottles segmented at 0.5 level of detection minimum confidence. The probable reason for the undetected bottled could be the various threshold values set at different stages of the model pipeline. In the list of configuration set for model training the number of proposals that the model gets post NMS suppression currently set at 2000 for training and 1000 for the inference. The NMS Algorithm1 shows how the proposals below threshold are suppressed for the next stage of processing. Changing the proposal thresholds parameter may improve the model performance.

5.4 Discussion

Instance segmentation of plastic waste bottles using transfer learning approach, for instance, segmentation performed for single object class with varied features shape features of waste bottles in public places using state of the art, Mask RCNN algorithm. In this work, custom dataset created from internet images to observe the customized model performance.

The Overall all the model generalised well at training phase.Though the models suffered at loss at validation stage due to the inadequate training and the threshold limitations set for the model pipeline the incremental fine tuning improved the performance at the validation stage. The Models performed quite well will the stage two transfer learning approach. The selective data augmentation of horizontal vertical flip at stage2 fine tuning did not contribute much to the improvement compared to the No augmentation training. ALL layers tuned at stage 2 with augmentation and without augmentation for 30 epochs achieved mAP0.55 and 0.56 respectively for Model, Phase2 RefNo_3. The models tuned for 4+ layers onwards performed better than the complete ALL layers tuning. Comparatively all the stage2 and stage3 models performed well above mAP of 0.55, however the highest precision of 0.59 achieved by model, 2SMODEL_ OS6.

The main challenges faced for this work was limited resource availability and the dataset. In order to overcome this, an optimal workflow designed, to run the model in a phased manner using the Transfer learning approach of fine-tuning the Convolutional Neural network. Since the custom dataset had limited images, the selective Data augmentation method applied to enhance the training capabilities of the Model. The model comparison with the varied experimental setup as discussed in the result section shows that with incremental training and the domain-specific dataset it has the potential to improve instance segmentation.

With the limited dataset, the model performed well when compared against the coco Benchmark evaluation method of Mean Average Precision. It is interesting to note how at varied incremental training the model performed. With deeper network backbone architecture, particularly the ResNet101, the instance segmentation performance improved and detection precision moved closer to human vision. It achieved mAP[0.5:0.95:0.5] over 99% accuracy on the training dataset and 59% accuracy on validation dataset.

Notably, the model performance is influenced by the number of images trained and also the resolution. Since in this work, images of varied size and dimensions resized to be fed into the network, and in order to maintain the model performance of feature extraction, the image dimension kept with a resolution of 1024×1024. However, this had resources allocation problems during the training phase that required hyper-parameter tuning of reducing the batch size and the learning rate parameter for the tensor to allocate the GPU resource.

The Backbone Feature training on Nvidia-GeForce RTX could not run the training due to its memory limitations. The Tesla V100 performance for image processing worked well with memory allocation problems at various instances. The model pipeline designing is one of the important tasks when the memory bottleneck issue occurs for training and evaluation process.

The shape feature of the bottles did affect the model performance as it could not predict deformed shape object good prediction confidence score. This problem needs to be addressed by increasing the training dataset, which will be the future work. However, the representa-

tional custom dataset performed well with the limited training due to resource constraint. It will be a good lead for the future work to improve on the training dataset availability and the use of the GAN-generative adversarial networks potential to create enough data for training. New techniques in deep learning broaden the scope of the problems addressed. Though the stage2 Model performed well in achieving a mAP of 59% , this work can be compared with other segmentation algorithms.

This work will be useful to be adopted for waste detection in public places. Though the work is restricted to a single class of waste bottle detection, it can be extended to other domain-specific implementations. Currently, the waste plastic bottle pollution problem that is under research for rivers and ocean clean-up will benefit from this work.

5.5 Summary

Phase 1 Model 101SGD performed well with least Mask RCNN loss of 3.81%Ḋeeper network-ResNet 101 with Stochastic gradient descent optimizer performs better compared to the shallow network ResNet50.

Phase 2 Model-2SMODEL_ 0S06 achieved mAP50 of 74% accuracy and mAP [0.5 : 0.05 : 0.95] 59% precision.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, the instance segmentation method of the Mask RCNN applied to the waste bottle object detection in images and video inferences to locate the object instances and provide the detected. Extensive experiments conducted with the custom dataset to analyse the performance of the algorithm under various parameter optimization to check the optimal performance of the model under limited resource availability and the dataset for prediction. The transfer learning techniques applied to ensure that the model performs well with the state of art technique. The data augmentation techniques applied to check the performance of the results against the normal training and fine-tuning technique. In this the custom dataset trained in three different phases to check on the prediction accuracy to find the optimal output. The techniques applied were layer heads training and fine tuning using selected layer training and/or all layers training with and without augmentation at different phases to compare the outcomes.

- Transfer Learning from pre-trained knowledge substantially reduces the time to train custom dataset.
- Mask RCNN applied to the waste bottle instance segmentation in images and video to locate the object.
- Augmentation techniques applied to analyse the model performance with limited dataset.
- Custom dataset transfer learning with initial head-layers training with ResNet101 architecture with fine tuning layers resulted in the mAP50 of 74% precision.

Contribution

Developed a segmentation model for bottle segmentation.

Analysed the model performance for the various transfer learning approaches adapted.

Analysed the strength of the algorithm over several bulk bottle instances.

Provide an available bottle segmentation framework and trained model.

6.2 Future work

Improvements in precision of detection could be achieved by training with additional dataset and modifying the training and inference network feeds.

Dense clustered bottle segmentation where the model ignores object segmentation can be focused to improve performance by changing the thresholds at training.

This implementation may be considered to implement for a more robust and powerful model development for detecting waste bottles in public places.

Single Stage Detectors such as the YOLO or the SSD architecture can be applied to compare its performance and could be studied where speed is the criterion.

The work may be extended to segment other class of objects causing environmental pollution.

Chapter 7

Reflection

- M.Sc dissertation work gave me the opportunity to explore and understand how various global organizations such as Google and Facebook invest in development of Machine Learning algorithms.
- Training Deep learning algorithms require huge computation resource both in terms of memory and processing power. Proper planning is required to productively use the resources.
- Step by step approach in training Deep networks to be followed to understand the complexities of processing as it is prone to memory leaks and disk space issue leading to code break-up issues and halt your work.
- Since most of the frameworks used in the study are still in research phase, it has taken a considerable amount of effort to find a working set of frameworks/libraries that are compatible to run the models.
- Project Management was a good learning experience right from the initial phase of research question and through the execution phase. MS-Office connections were very good, and it helped in effective project communications. However, remote working, resource constraint, memory leakage, home internet connectivity issues and code breakup had a great impact on smooth execution of the project work. It was a challenge to get access to the shared GPU as it was quite busy.

References

Abdulla, W. (2017), 'Mask r-cnn for object detection and instance segmentation on keras and tensorflow https://github. com/matterport', *Mask _ RCNN* .

Brownlee, J. (2017), 'A gentle introduction to transfer learning for deep learning'.
URL: *https://machinelearningmastery.com/transfer-learning-for-deep-learning/*

Brownlee, J. (2019), 'A gentle introduction to object recognition with deep learning'.
URL: *https://machinelearningmastery.com/object-recognition-with-deep-learning/*

Chollet, F. (2015), 'others. 2015', *Keras: Deep learning library for theano and tensorflow. URL: https://keras. io/k* .

Deng, J., Dong, W., Socher, R., Li, L., Kai Li and Li Fei-Fei (2009), Imagenet: A large-scale hierarchical image database, *in* '2009 IEEE Conference on Computer Vision and Pattern Recognition', pp. 248–255.

Everingham, M. and Winn, J. (2010), 'The pascal visual object classes challenge 2010 (voc2010) development kit'.
URL: *http://host.robots.ox.ac.uk/pascal/VOC/voc2010/devkit*$_d oc_0 8 - May - 2010.pdf$

Fulton, M., Hong, J., Islam, M. J. and Sattar, J. (2019), 'Robotic detection of marine litter using deep visual detection models', *2019 International Conference on Robotics and Automation (ICRA)* .
URL: *http://dx.doi.org/10.1109/ICRA.2019.8793975*

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V. and Garcia-Rodriguez, J. (2017), 'A review on deep learning techniques applied to semantic segmentation', *arXiv preprint arXiv:1704.06857* .

Geron, A. (2017), *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, Sebastopol, CA.

Girshick, R. (2015), Fast r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 580–587.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Hariharan, B., Arbeláez, P., Girshick, R. B. and Malik, J. (2015), 'Hypercolumns for object segmentation and fine-grained localization', *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 447–456.

Hariharan, B., Arbeláez, P., Girshick, R. and Malik, J. (2014), 'Simultaneous detection and segmentation', pp. 297–312.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017), Mask r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 2961–2969.

He, K., Zhang, X., Ren, S. and Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. et al. (2017), Speed/accuracy trade-offs for modern convolutional object detectors, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 7310–7311.

Ioffe, S. and Szegedy, C. (2015), 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint arXiv:1502.03167* .

Kambam, L. R. and Aarthi, R. (2019), Classification of plastic bottles based on visual and physical features for waste management, *in* '2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)', IEEE, pp. 1–6.

Kokoulin, A. N., Tur, A. I. and Yuzhakov, A. A. (2018), Convolutional neural networks application in plastic waste recognition and sorting, *in* '2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)', IEEE, pp. 1094–1098.

Lin, M., Chen, Q. and Yan, S. (2013), 'Network in network'.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017), Feature pyramid networks for object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2117–2125.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L. (2014), Microsoft coco: Common objects in context, *in* 'European conference on computer vision', Springer, pp. 740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016), 'Ssd: Single shot multibox detector', *Lecture Notes in Computer Science* p. 21–37.
URL: *http://dx.doi.org /10.1007/978-3-319-46448-0*$_2$

Mittal, G., Yagnik, K. B., Garg, M. and Krishnan, N. C. (2016), Spotgarbage: smartphone app to detect garbage using deep learning, *in* 'Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing', pp. 940–945.

Parker, L. (2019), 'How the plastic bottle went from convenience to curse', `https://www.nationalgeographic.co.uk/environment-and-conservation/2019/08/how-plastic-bottle-went-miracle-container-despised-villain`.

Parthasarathy, D. (2017), 'A brief history of cnns in image segmentation: From r-cnn to mask r-cnn', *Available online: blog. athelas. com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-rcnn-34ea83205de4* .

Proença, P. F. and Simões, P. (2020), 'Taco: Trash annotations in context for litter detection', *arXiv preprint arXiv:2003.06975* .

Rad, M. S., von Kaenel, A., Droux, A., Tieche, F., Ouerhani, N., Ekenel, H. K. and Thiran, J.-P. (2017), A computer vision system to localize and classify wastes on the streets, *in* 'International Conference on computer vision systems', Springer, pp. 195–204.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779–788.

Ren, S., He, K., Girshick, R. and Sun, J. (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, *in* 'Advances in neural information processing systems', pp. 91–99.

Shorten, C., K. T. (2019), 'A survey on image data augmentation for deep learning'.
URL: *https://doi.org/10.1186/s40537-019-0197-0*

Soviany, P. and Ionescu, R. T. (2018), 'Optimizing the trade-off between single-stage and two-stage object detectors using image difficulty prediction'.

Sumit, S. (2019), 'Exploring data augmentation with keras and tensorflow'.
URL: *https://towardsdatascience.com/exploring-image-data-augmentation-with-keras-and-tensorflow-a8162d89b844*

Torrey, L. and Shavlik, J. (2010), Transfer learning, *in* 'Handbook of research on machine learning applications and trends: algorithms, methods, and techniques', IGI global, pp. 242–264.

Wang, J., Guo, W., Pan, T., Yu, H., Duan, L. and Yang, W. (2018), Bottle detection in the wild using low-altitude unmanned aerial vehicles, *in* '2018 21st International Conference on Information Fusion (FUSION)', IEEE, pp. 439–444.

Weng, L. (2017), 'Object detection for dummies part 3: R-cnn family', *lilianweng.github.io/lil-log* .
URL: *http://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html*

Yang, M. and Thung, G. (2016), 'Classification of trash for recyclability status', *CS229 Project Report* **2016**.

Zhao, Z.-Q., Zheng, P., Xu, S.-t. and Wu, X. (2019), 'Object detection with deep learning: A review', *IEEE transactions on neural networks and learning systems* **30**(11), 3212–3232.