Exploring Bounding Box and Instance Segmentation to Detect and Classify Garments.

Bidish Basu
School of Computing
Dublin City University, Ireland
Email: bidish.basu2@mail.dcu.ie

Sachin Mahesh
School of Computing
Dublin City University, Ireland
Email: sachin.mahesh3@mail.dcu.ie

***Abstract*—Fashion is a vital part of our lives. Detecting and classifying clothing items presents a significant challenge in today's world, where fashion is growing and changing at a rapid pace. Through our paper, we aim to find how accurately we can detect clothing items in images and make further predictions. We implement two methods, bounding box and instance segmentation for the analysis. We aim to find out how each of these detection algorithms performs on images. Additionally, we use the output of the bounding box and use it as the input to the segmentation and see if there are any improvements in the detection. The paper proposes to measure the prediction accuracy using the Intersection Over Union and the Mean Average Precision metrics and hence compare the results to observe which method is more effective in classifying and predicting the clothing items. Further, we determine the significant pixels which the model uses to detect and classify the objects within an image. Our work indicates that using the original images (40.4%) gives a slightly better score than using the cropped images (34.3%) when measured using the mean average precision metric; which is an indication that the background affects the detection and classification of objects in an image and cropping an image to omit the background does not improve the accuracy.**

***Index Terms*—Bounding Box, Instance Segmentation, Intersection Over Union, Mean Average Precision, Mask R-CNN, YOLO v3, Saliency Map.**

I. Introduction

The study of detection and classification of garments has continually been an active topic because of its vast potential to tap into a billion-dollar industry. There has been a substantial amount of work accomplished in the domain with the development of state-of-the-art techniques like neural networks and deep learning algorithms. However, extracting valuable information from fashion images in real-world applications possesses a unique challenge. There is a vast diversity in the types of garments worn and also as garments can be of various shapes and sizes. There are many datasets developed with the intention of clothing detection. Two prevalent datasets used for the analysis in this paper are DeepFashion [12] and DeepFashion2 [6], [7].

A widely used object detection technique is the Bounding Box implementation [10], a technique where a rectangle encloses an object in a 2-dimensional space. Many techniques have been developed to find the best-fit bounding box. One such method is YOLO [15] (You Only Look Once), which uses a custom variant of DarkNet-53, which contains a pre-trained network of 53 layers based on ImageNet [5].

Fig. 1: Outputs of Bounding Box using YOLO-v3 and Instance segmentation using Mask R-CNN.

These 53 layers are stacked on each other to form a 106 fully convolutional layered architecture. It detects an object at three different scales and uses feature or highlight map from previous layers for detection. In order to perform object detection, feature maps are collected from three different points in the network, and a 1x1 detection kernel is applied to get the final class value. [15]

Furthermore, advancements have been made in object detection and semantic segmentation. For object detection, there are systems such as Faster R-CNN [16], and for semantic segmentation, Fully Convolutional Network (FCN) [13].

Faster R-CNN takes the image input and returns a feature map, on which a Region Proposal Network (RPN) is applied to get the candidate bounding boxes. These proposed outputs from the RPN, after passing through an RoI pooling layers are passed into a Fully Convolutional Network (FCN) to classify and output the bounding boxes enclosing the objects. [9], [16]. We can achieve semantic segmentation using a Fully Convolutional Network (FCN) [9] [13]. All the layers in an FCN are convolutional, and an FCN uses the convolutional layers to classify each pixel in an image. In the final output, it highlights each pixel in a colour that corresponds to the

Fig. 2: The first image is the output of Mask R-CNN on the original image. The second image is the output of Mask R-CNN on the cropped image. The cropped image is the result of the output from the YOLO model.

classes used for segmentation.

In order to detect individual items of clothing in an image, we use Mask R-CNN [9], an Instance Segmentation framework in our work. Semantic segmentation classifies multiple objects of a class as a single entity, whereas instance segmentation classifies multiple objects of a class as an individual instance of the class. Instance segmentation in Mask R-CNN combines the two tasks of object detection and semantic segmentation. Mask R-CNN has the same first step as Faster R-CNN, and in the second step, along with predicting the class of the object, Mask R-CNN also calculates a binary mask for each object present in an image [9].

We divided the task of image detection and classification into two stages. In the first stage, both the bounding box and instance segmentation were performed (Figure 1). For performing detection using the bounding box, we used the YOLO framework on the DeepFashion dataset, and for instance segmentation, using the Mask R-CNN framework, we used the DeepFashion2 dataset. We measure the performances of both the models using the Intersection Over Union (IOU) and Mean Average Precision (mAP) metrics. In the second stage, the output of the YOLO model, i.e., the object contained within each of the bounding boxes are cropped from the original image and used as an input to the Mask R-CNN (Figure 2). We then compare the performances in both the stages and report our findings.

Multiple factors affect the detection of objects in an image. Through our work, we try to determine whether or not the background plays a role in detecting objects in an image.

II. LITERATURE REVIEW

DeepFashion: The DeepFashion dataset contains images that are annotated with multiple attributes, clothing landmarks, and correspondence of images. The images are aggregated under different scenarios, which help in the development of fashion detection and classification. The paper also proposes a deep learning model called FashionNet. FashionNet predicts clothing landmarks and clothing attributes jointly and learns in the process [12].

DeepFashion2: The DeepFashion2 dataset overcomes the limitations of the DeepFashion dataset, which includes a single clothing-item per image, sparse landmarks, and no per-pixel masks, which affect the broader usage of the dataset. DeepFashion2 has much more extensive annotations than the DeepFashion dataset, and each image has annotations such as style, scale, mask, bounding box, occlusion, viewpoint, and dense landmarks. These annotations put together makes DeepFashion2 a more versatile dataset [6], [7].

Edge Boxes: Locating Object Proposals from Edges [19] proposes a method for generating bounding boxes around objects using edges. The paper proposes generating object bounding box using the number of contours that are wholly contained within a bounding box. The paper suggests a straightforward scoring function that computes the weighted sum of the edge strengths within a box, minus those that are part of a contour that straddles the box's boundary. This is done to prove that, scoring a bounding box based on the number of contours it encloses, creates an efficient model in detecting objects, rather than just counting the number of edge pixels within the box. Two aspects of the predicted annotations are measured, namely, quality, using IOU, and mAP.

Classification of Fashion Article Images using Convolutional Neural Networks: [2] This paper utilizes a two-layer CNN approach with batch standardization and skips connection architecture to classify images from the fashion dataset using neural network-based deep learning. This is achieved by creating a feature map of the input images. The more significant part of the neural system is prepared using backpropagation, and the principle boundary in this model is a multi-layer perceptron network used for characterizing images. Further, overfitting of the model is controlled by using drop out techniques. Object recognition accuracy is effectively increased by training the network on a broad set of bounding box annotations. A part of the deep learning networks is pre-trained as a classifier for object detection. The parameters are updated by utilizing stochastic gradient descent and backpropagation methods [4]. Most of their pre-processing is based on region proposal method and adopted multi-scale combinatorial grouping and greedy iterative solution to find the local optima.

YOLO-v3: This framework is an improvement over the initial YOLO version [14]. YOLO-v3 [15] uses an advanced neural network and anchor boxes to predict the dimensions of the bounding box. The best bounding boxes are chosen depending on the area of the image that the predicted box covers over the ground truth box, which surpasses a threshold value, and the rest are disregarded. Class predictions are implemented using multi-label classification for the objects encased within the box based on independent logistic classifiers, and during the training process, a binary cross-entropy loss is used for the class predictions.

Fast R-CNN: Fast R-CNN builds on the traditional R-CNN. Fast R-CNN uses a single model to extract features from Regions of Interest (RoI) and then divides these extracted features into multiple classes. Simultaneously, it returns bounding

boxes that enclose the objects in the image of the classes generated. Fast R-CNN takes an image as input, which is then passed into a ConvNet. This ConvNet then generates the RoI in the image, and then an RoI pooling layer is applied to these regions. This reshapes the regions according to the input to ConvNet. Further, the regions are passed on to an FCN [13] on top of which a Softmax layer is applied to identify the classes. Furthermore, the bounding box coordinates for the predicted classes are found out using a linear regression layer used in parallel [8].

Faster R-CNN: Faster R-CNN builds on the previous Fast R-CNN. It contains two steps. In the first step, the image is passed into a ConvNet from which a feature map is generated. Then the candidate bounding boxes are determined using a Region Proposal Network (RPN), which contains the objects being detected. The second step is to extract the features present in each of these bounding boxes and perform classification and bounding box regression. In the second step, the features are extracted using a RoIPool, as seen in Fast R-CNN [9], [16].

Mask R-CNN: Mask R-CNN is an instance segmentation framework that classifies multiple objects of a class as an individual instance of the class. Mask R-CNN takes the output of the Faster R-CNN, which are the class labels and the predicted bounding box, and draws a binary mask within the bounding box. The pixel-wise binary mask indicates the location of the object being detected within the bounding box. This binary mask is built using the Fully Convolutional Network [9].

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps: The paper suggests two visualization techniques, one to generate a substitute image which identifies the targeted classes. The other is to create a Saliency Map that highlights the part of an image that describes the classes. The paper shows that the Saliency Maps generated can be used with weakly supervised object segmentation models [17].

III. DATA PREPROCESSING

A. YOLO

We train the YOLO-v3 model on the DeepFashion dataset, which comprises 50 classes. We completed an initial exploratory data analysis and discovered that the dataset had an unbalanced number of instances across various classes. Consequently, few classes had barely any images in it, hence we worked with the top ten classes with the most noteworthy number of pictures. Further, to adjust the dataset, we performed image augmentation on the top ten classes; for example, contrast changes, the addition of Gaussian noise, brightness variation and image sharpening to accomplish the consistent number of images for each class to maintain a uniform distribution between the classes for further processing.

Once we had generated the augmented images, we required the annotations to process the dataset further. We also needed to ensure that the orientations of the pictures did not change during the image augmentation process as we wanted to use the same ground truth annotation for training. We extracted the initial bounding box values from the underlying dataset. However, YOLO-v3 has a different approach for annotations as opposed to the customary (x1, y1, x2, y2) values. We had to change these values according to the image width and height, such that, when we pass the input through the network (with various down-sampling and up-sampling techniques), relative estimations of the bounding box remained the same.

B. Mask R-CNN

The dataset used for Mask R-CNN is DeepFashion2. The DeepFashion2 is composed of 13 classes. In order to prepare the data for the analysis, we first inspected the dataset for any anomalies. The dataset did not contain an even distribution of data among all the classes, similar to DeepFashion. To have a uniform distribution of data across the classes and to prevent the loss of data points, we merged some classes and formed new classes. For example, the original dataset featured Short Sleeve Dress, Long Sleeve Dress, Vest Dress, and Sling Dress as four different classes. These classes did not have many data points for training. To make the number of training data points uniform, we merged these classes into a single class called Dress. Similarly, other classes that had lesser training points were combined, and finally, we formed six classes. The classes which we finally used for the analysis are Short Sleeve Top, Long Sleeve Top, Skirt, Shorts, Trousers, and Dress.

DeepFashion2 has a separate annotation file for each image. In order to make sure that the annotations followed the Mask-RCNN implementation [1], we made changes to the annotation files. The first step to prepare the final annotation file was to extract only the information needed for implementing the instance segmentation. Since Mask-RCNN was used only for the segmentation part of the analysis, we extracted the data present in the segmentation section of each of the annotation files. DeepFashion2 provides us with information about multiple objects belonging to various classes in an image. We then extracted the classes and their corresponding segmentation ground truth values from each annotation file and converted them into the format suitable for the implementation. Once we had extracted all the classes and the segmentation values, we combined the annotation information into a single file that contained only the class names and the segmentation points of all the images in the dataset. We performed these steps separately for both the training set and validation set.

C. Cropped Images

To generate cropped images, we passed images from the DeepFashion2 dataset into the YOLO-v3 model. Next, we generated the respective bounding boxes and cropped the images as per the generated bounding boxes. We then passed the cropped images into the Mask R-CNN model for the final analysis.

IV. IMPLEMENTATION

A. YOLO

YOLO-v3 utilizes a custom deep learning architecture, DarkNet-53, which is trained on an image database sorted out

as per WordNet. Thousands of images represent every stage of the hierarchy in the structure [5].

For object detection, we stack 53 additional layers on top of one another, which creates 106 fully connected convolutional layers. Detection is done at three different scales at 79, 91, and 106, with strides shifting from 32, 16, and 8. We have down-sampled the input pictures into 416x416 dimensions, which decide the stride in the system. Besides, predictions at these three scales are acquired by downsampling the input images into aspects of 32, 16, and 8. We chose these dimensions for the input images in order to supplement the generation of anchor box down the pipeline.

We have implemented YOLO-v3 on Google Colaboratory (Colab) with parameters dependent on the original paper [15] and fine-tuned some of the parameters in order to work on the DeepFashion dataset. YOLO tries predicting various bounding boxes for each picture and characterizes them depending on the attributes and images which cross the threshold. To extend the performance of the model, we needed to change a handful of parameters. We generated the anchor box values using the k-means clustering algorithm on standardized height and width of the ground truth bounding boxes from the initial dataset, K being 9 in this scenario, i.e., three anchors for each scale.

We began with a batch size of 128, but the training loss did not reduce as much as we had anticipated. Hence, we decided on a batch size of 64 images with labels in the forward pass to compute the gradient value and update the weights through backpropagation. We partitioned each batch into eight blocks, which ran parallelly on the GPU. We normalized the input images to 416x416 dimensions containing three channels.

We used a weight decay regularization of 0.0005 to keep the weights from growing too large. We utilize a momentum of 0.9, which connotes the number of future weights is affected by past weights. We use a learning rate of 0.001 with a burn-in value of 1000, i.e., the learning rate increases exponentially from 0 to 0.001 over the initial 1000 iterations. We update the learning rate by multiplying with a scale estimation of 0.1, during training, when the total number of processed batch reaches 80% and 90%, respectively.

We evaluate the model using multiple metrics. The first, we compute the Intersection Over Union (IOU), by dividing the area of intersection over the area of union between the ground truth bounding box and the predicted bounding box. Consequently, the IOU ranges from 0 to 1. We designed the model to predict a bounding box, such that, the IOU score is close to 1.

$$IOU = \frac{Area\ of\ Intersection}{Area\ of\ Union}$$

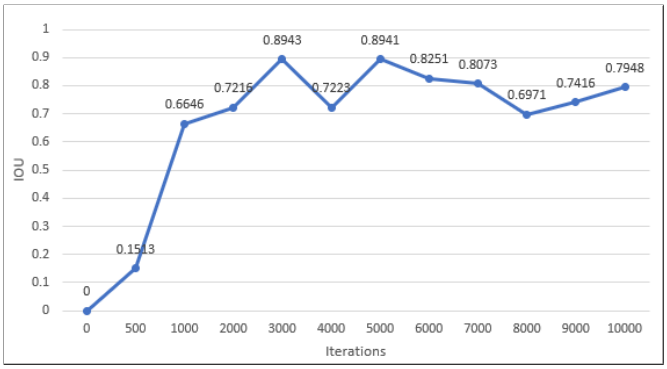

Fig. 3: YOLO-v3: IOU accuracy of a single image in the DeepFashion dataset over 10000 iterations.

Figure 3 depicts the IOU accuracy of a single image passed through the entire network, and the value keeps oscillating back and forth over 80%.

Next, we calculate the Average Precision (AP) and Mean Average Precision (mAP). To calculate these metrics, we need Precision and Recall. We determine precision by finding the proportion between True Positive (correct predictions) and total predictions (correct predictions and false predictions). We determine Recall by finding the ratio between True Positive and the total number of items in the dataset, i.e., true positive and false negatives.

$$Precision = \frac{TP}{TP + FP} \quad \text{and} \quad Recall = \frac{TP}{TP + FN}$$

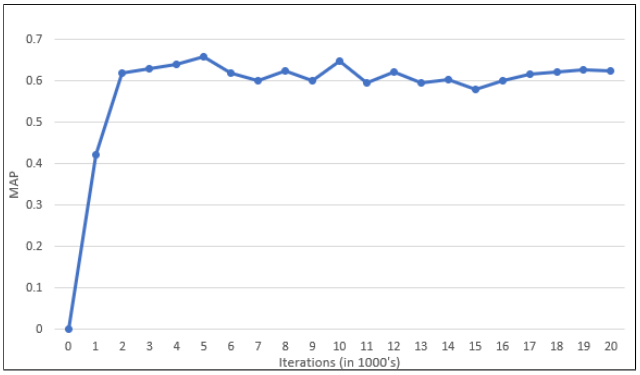

Fig. 4: YOLO-v3: Progression of mAP over 20000 iterations.

We calculate AP by finding the region under the accuracy-recall curve. AP is a combination of Recall and Precision, and mAP is the mean of the AP scores determined for each class. Figure 4 portrays the Mean Average Precision value when processed through the neural network for over 20000 iterations, the mAP value shoots up during the intial stage, and then peaks at 5000 and 10000, finally it balances out as the training procedure proceeds.

We set the maximum number of batches according to the top 10 classes and changed the steps to 80% and 90% of maximum batches to fine-tune the model for both training and testing configurations.

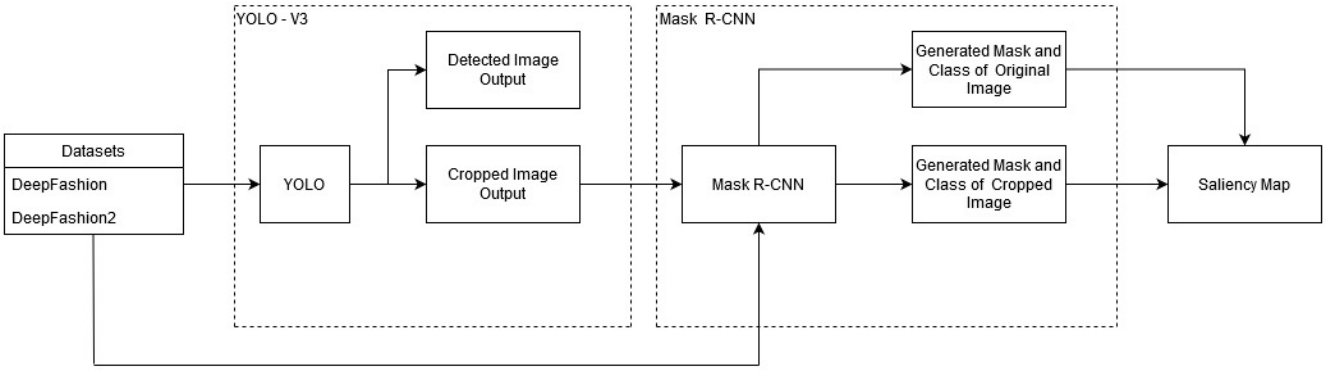

Fig. 5: YOLO-v3 and Mask R-CNN frameworks used for generating bounding boxes and instance masks on the DeepFashion and DeepFashion2 datasets in addition to creating Saliency Maps.

The following equation altered filters in the YOLO layer.

$$Filter = (num\_classes + 5) * 3$$

During the testing phase, we changed the batch size and subdivisions to 1, and the rest of the configuration was the same as the training phase. We trained the model up to 20000 iterations and recorded the following scores by running against the test set images.

Iteration	Precision	Recall	F1 Score	mAP	Avg IOU
2000	0.55	0.59	0.57	61.98 %	42.73%
4000	0.55	0.63	0.59	63.98 %	44.15 %
6000	0.53	0.62	0.57	61.91 %	43.65 %
8000	0.55	0.62	0.58	62.33 %	45.09 %
10000	0.55	0.62	0.59	64.74 %	45.60 %
12000	0.57	0.63	0.60	62.11 %	47.07 %
14000	0.55	0.62	0.58	60.35 %	44.28 %
16000	0.56	0.64	0.60	60.18 %	46.26 %
18000	0.60	0.65	0.62	62.09 %	50.86 %
20000	0.58	0.64	0.61	62.45 %	50.15 %

TABLE I: Performance evaluation of the YOLO-v3 model on the DeepFashion dataset.

Table I gives a rundown of the YOLO-v3 model with respect to iterations, Maximum Precision, Recall, F1 Score and Average IOU. We achieved a maximum Precision value of 0.60 at 18000 iterations, and the average IOU peaked at the same mark. The mAP value was oscillating and peaked at the 10000th iteration.

Class predictions utilize a multi-label approach, i.e., independent logistic classifiers and binary entropy loss functions rather than softmax activation function, to avoid overlapping labels in multi-label classification. Softmax for class expectation forces the assumption that each box has precisely one class, which is often not the case in the dataset. For instance, an individual wearing a shirt and a jacket will have a single bounding box with various labels attached to it.

B. Mask R-CNN

To detect and classify images, we first divide the image into multiple segments. It is computationally expensive to process the entire image, pixel by pixel, as there are regions within the image that may not contain any useful information. By segmenting the image, we make the processing of the image easier and reduce the time taken to detect and classify objects within images.

The two most commonly used techniques for image segmentation are semantic segmentation and instance segmentation. Semantic segmentation classifies multiple objects of the same class as a single instance. In contrast, instance segmentation will classify objects of each class as individual elements. For example, if we want to detect only shirts in an image, semantic segmentation will classify all the detected shirts in an image as a single instance of the class shirts. However, instance segmentation will identify each of these shirts individually.

We use Mask R-CNN to implement instance segmentation in our work. Mask R-CNN is an instance segmentation technique that extends the working of Faster R-CNN, which is used for object detection and generating bounding boxes around the detected objects, by adding a binary mask around each of the objects detected within an image [9], [16].

The backbone of Mask R-CNN is the ResNet-101 architecture. We again used the Colab to implement the Mask R-CNN model. Initially, we implemented Mask R-CNN with the parameters as per the original paper [9] on the DeepFashion2 dataset with 13 classes. We then changed the hyperparameters following the existing implementation of Mask R-CNN [1], [9] as needed. We trained the model on the DeepFashion 2 dataset using the 12 GB GPU provided within Colab. We set the maximum image size to 1024x1024, which allowed us to fit two images per GPU for training. As per the existing implementation [1], a 12 GB GPU can typically accommodate a maximum of 2 images of size 1024x1024. Since the dataset had images of various sizes and larger images would lead to higher computational time, we enabled mini-mask and used a mini-mask of size 56x56. During the training process, the model generates binary masks. If we use images of higher resolution, then the masks generated will require an ample memory space. Using mini-masks helped resize the instance masks, which in-turn helped reduced the load on the GPU and

the memory consumed.

We set the number of ROIs generated per image to 200, as this helped improve the training speed. DeepFashion2 has maximum ground truth values of 10 garments per image; hence, for training, we set the maximum number of ground truth instances to 10 per image. Similarly, for the number of final detections, we set the maximum instances detected per image to 10. We set the learning rate initially to 0.02 as per the original paper [9], but TensorFlow caused the weights to destabilize, and the loss increased after running a few epochs. To address this, we used a learning rate of 0.001, for the initial 200 epochs and reduced it by 10, as we further trained the model, which further helped decrease the loss. We kept the learning momentum as per the paper at 0.9 and set the weight decay to 0.0001. We set the minimum confidence to 0.9, which meant that we use only those detections with a minimum confidence of 90% for training.

For the final model, we used six classes, which resulted from pre-processing the dataset, for differentiating between the types of garments. The classes which we used for the final analysis were Short Sleeve Top, Long Sleeve Top, Skirt, Shorts, Trousers, and Dress. Since we were only interested in instance segmentation using the DeepFashion2 dataset, we did not consider the bounding box ground truth values for training. Instead, we generate the bounding boxes by picking the smallest box that would envelop the generated masks [1], which helped us calculate the average precision of the masks generated alone. To measure the performance per class, we used the ground truth and predicted values of both the bounding box and the segmentation masks. We measured the performance by calculating the average precision values of the generated masks of each class (mask AP), and further by calculating the mAP. We measure mask mAP values at IOU of 50% and over, and IOU of 75% and over, indicated by mAP_{50} and mAP_{75}, respectively.

To check whether the background had any influence on object detection, we generated Saliency Maps [17] across the classes, as seen in Figure 6.

C. Cropped Images

We then passed the images from the DeepFashion2 dataset into the YOLO model as seen in Figure 5, and the images were cropped based on the bounding boxes detected using the model. To crop the input images based on the predicted bounding box coordinates, we had to alter the source code before building the DarkNet. These coordinates were utilized on the original images and edited as follows; we updated the X and Y estimations of the upper left corner by adding the difference in x and y values of the bounding box to the original value, height and width of the image were determined by expelling the margins from the original image. We then passed these cropped images to the final Mask R-CNN model and then tested the output. We compared the mask AP and the mAP of each of the six classes with the original images.

D. Saliency Map

We generate the Saliency Maps [17] with the final weights from the Mask R-CNN model for both the original and cropped images from the DeepFashion2 dataset as depicted in Figure 5. We use the Saliency Maps to identify the pixels in the images that are most significant and help detect and classify the objects within images. These maps help us visualize how the model decides, i.e., if the model makes a prediction based on the object or does it use the environment to perform classification.

V. EXPERIMENTATION AND RESULTS

We conducted multiple experiments to check if the background influenced the detection of garments. First, we implemented object detection utilizing the bounding box approach using YOLO-v3 with DarkNet-53 as its framework. We created diverse test sets to obtain the average precision values for each class, which are tabulated in Table II.

Class Name	DeepFashion	DeepFashion2
Dress	84.77	55.81
Long Sleeve Top	58.25	48.79
Short Sleeve Top	41.48	17.79
Shorts	51.4	26.93
Skirt	77.53	49.54
Trousers	48.52	42.63
Mean AP	60.33	40.20

TABLE II: YOLO-v3: Average Precision values of DeepFashion and DeepFashion2.

We get the mAP estimation of 60.33 for DeepFashion and 40.2 for DeepFashion2. In DeepFashion, most of the images contain balanced lighting with a bright solid coloured background. In contrast, most images in DeepFashion2 include a busy background (outdoor setup or varying capture angles), this may imply that several other factors affect the detection and classification process, along with the objects in the image.

Dataset	Backbone	mAP_{50}	mAP_{75}
DeepFashion2 with 13 classes.	ResNet-101	21.4	15.2
DeepFashion2 with 6 classes.	ResNet-101	40.4	22.6

TABLE III: Mask R-CNN: A comparison of the mask mAP values of the DeepFashion2 dataset between the original 13 categories of garments, and a reduced number of 6 categories at IOU 50% and 75%.

Next, we conducted experiments on the DeepFashion2 dataset. The implementation of Mask R-CNN [1] supports ResNet-50 and ResNet-101. Since the current implementation of Mask R-CNN [1], [6] performs better on the ResNet-101 architecture, we use that as the backbone in our work. Initially, we executed Mask R-CNN on the DeepFashion2 dataset with 13 classes. We have tabulated the results of the same in Table III. We see that the mAP_{50} and mAP_{75} values are 21.4 and 15.2, respectively, for the original dataset with 13 classes. On exploring the dataset [6], we saw a vast difference in the number of images available to train across all the categories. To address this challenge, we combined multiple categories under different umbrella categories and came up with six final categories. We executed Mask R-CNN on the updated dataset

Categories	Dress Original	Dress Cropped	Long Sleeve Top Original	Long Sleeve Top Cropped	Short Sleeve Top Original	Short Sleeve Top Cropped	Shorts Original	Shorts Cropped	Skirt Original	Skirt Cropped	Trousers Original	Trousers Cropped
mAP_{50}	59.1	56.4	38.7	35.5	59.2	52.4	23.7	17.2	33.4	26.4	28.1	17.9
mAP_{75}	54.4	51.5	12.6	10.9	34.6	30.4	5.4	4.6	27.4	10.4	1.4	0.8

TABLE IV: Mask R-CNN: Mask mAP scores of the original and cropped images of 6 classes on the DeepFashion2 dataset at IOU 50% and 75%.

and saw tremendous improvement in the performance. Table III shows the mAP_{50} and mAP_{75} values as 40.4 and 22.6, respectively, for the modified dataset.

Images	Backbone	mAP_{50}	mAP_{75}
Original	ResNet-101	40.4	22.6
Cropped	ResNet-101	34.3	18.1

TABLE V: Mask R-CNN: Aggregated mask mAP scores of the original and cropped images of 6 classes on the DeepFashion2 dataset at IOU 50% and 75%.

Subsequently, once we had the six categories, we wanted to see how well the model performed on each of these classes. Table IV and Table V show the performance of Mask R-CNN on the modified DeepFashion2 dataset.

From Table IV, we see that we get excellent scores for Dress and Short Sleeved Top, which is close to the ones posted in the original implementation of Mask R-CNN [1], [6]. Dress and Short Sleeve Top had comparatively more samples for training, and the ground truth values of the segmentation masks were more accurate compared to other categories, indicating the better performance of these two categories.

Additionally, Long Sleeve Top, Skirt, and Trousers had comparable scores. Upon further inspection of the predicted images, we saw that Shorts had the worst score, as the masks created by the model were very similar for Shorts and Skirt. Hence, the model incorrectly predicted the instances of the class Shorts as Skirt, and therefore, the performance reduces.

Finally, Table V shows us the performance of the whole dataset. To gather the final scores, we computed an average of the scores tabulated in Table IV.

We see from the tabulated results that the original images performed much better than the cropped images. In order to find out the pixels that were affecting the predictions, we generated Saliency Maps, as shown in Figure 6. In visual processing, Saliency indicates unique features in an image such as resolution or pixels in an image. These features describe the visually appealing areas in a picture, which helps in differentiating between various features present in an image. For instance, coloured images are converted to black and white, to break down the most grounded hues present in them. Different cases would utilize infrared to distinguish temperature (red shading is hot, and blue is cold) [17].

In Figure 6, we see there are two images. The images on the left are the predictions of the Mask R-CNN model on the original and cropped images, respectively. The images on the right are the Saliency Maps generated on the original and cropped images. The lighter colour indicates the significant pixels that are used by the model for detection and prediction

Fig. 6: The images on the left are the output of Mask R-CNN and the ones on the right are the heat maps, that indicate the significant pixels used for object detection.

of garments in an image. We see in both the original and the cropped images, most of the significant pixels, lie outside the actual object being detected. The Saliency Maps indicate that the background of an image influences the predictions of the objects in the image; hence we get a much better score on the original images compared to the cropped images, as seen from the tabulated results.

VI. DISCUSSION

As we began our work towards object detection and classification utilizing two unique techniques, bounding box and image segmentation, we were confronted with multiple obstacles. Throughout our work, we conducted various trials; while we did not get tremendous outcomes from all, some were worth noting.

A. Data Preprocessing

DeepFashion contains images with numerous attributes. We needed to sift through the attributes to meet our requirements. Most of the attributes were split among multiple files and had to be combined and then converted into the desired format for further processing. While reading the annotation files into the python environment, majority of the rows were getting overlooked because of the quote issue, as there is no standard for comma-separated value files. Subsequently, we addressed the line terminator as it was causing the problem. To do so, we replaced it with a newline character, and further, we

combined the files containing bounding box values of images and categories files using image path as a grapple point.

DeepFashion2, on the other hand, has a separate annotation file for each of the images. In order to prepare the dataset to work with Mask R-CNN, we had to combine all the annotation files into a single annotation file. DeepFashion2 has rich annotations [6]. Extracting only the necessary attributes was a challenging task, as we had to iterate over a large number of annotation files and extract only the ground truth values that were necessary for the implementation and combine them into a single annotation file.

B. Implementation

Our objective is to compare the object detections between the images provided in the original DeepFashion2 dataset with that of the cropped images, which we get as a result of processing the original images. In order to crop the images, we tried implementing multiple architectures on the DeepFashion dataset and migrated the best model to crop the images from the DeepFashion2 dataset. Initially, we began our work on the more straightforward DeepFashion dataset as we were interested in developing a way to crop the images based on the predicted bounding boxes. Once we had a way to crop the images, we migrated the code to DeepFashion2. We chose DeepFashion2 over DeepFashion because DeepFashion does not have the ground truth values for segmentation, which was the second part of our work.

The first method which we utilized to extract the bounding boxes was using Fastai architecture, as the dataset was in the COCO format, and Fastai had native support for this format. The model performed well for multi-label classification and integrating it to predict bounding boxes was simple as it was baked with data block API for object detection. Despite the simple architecture, training consumed a considerable amount of time and predicted scores were not at par, and hence we had to choose a different model.

Next, we tried implementing Tiny-YOLO, which uses Dark-Net architecture as assets restricted us regarding computation time and processing power. Tiny-YOLO performed better when contrasted with Fastai and had a superior training speed. Nevertheless, because of architectural drawback over the typical fully fledged YOLO architecture, as Fastai needed multiple passes for each image to train, its performance was limited and predicted scores were subpar.

We then looked into the implementation using YOLO-v3, which has an advanced architecture than its counterparts (Tiny-YOLO, YOLO-v2) and performed better. However, this was more resource-intensive than Tiny-YOLO and required more graphical processing power. When we tried running this on the GPU provided by the university, we often faced 'GPU out of memory error' because of limited memory. To overcome this, we tried changing the input images into greyscale and even reduced the batch sizes significantly. However, we could not make it work on the GPU provided. Subsequently, we turned to Colab, which provided us with a 12GB NVIDIA Tesla K80 GPU.

In the implementation of Mask R-CNN, like YOLO-v3, we faced more challenges. We faced problems similar to YOLO-v3 when we tried executing the model on the GPU provided by the university. Since we had a small batch size and a single GPU to work with, and a large number of images to iterate over; to compute a single epoch, it took a considerable amount of time. Since we had multiple experiments to run, we needed faster GPUs; hence we made use of Colab.

Colab presented further challenges. The first challenge we faced was with the dataset. Since the dataset was large, and we needed the data to be present within Colab to work on, we fragmented the dataset and used Google Drive as the storage. We then referenced the dataset into the Colab and worked on it. The next challenge was with the batch size. Since we had only a 12GB GPU, we could fit only a small number of images in the GPU, which considerably increased the training time. Colab has a period of 12 hours where we could continuously train, but once we reached the time limit, we had to wait for 24 hours after which we could continue to train. Due to this restriction, we trained the model continuously until the time limit and saved the trained weights. We used these saved weights to continue training.

C. Limitations

YOLO performs a significant number of mistakes compared to Faster RCNN; specifically location errors. On the other hand, YOLO performs fewer background errors than Faster RCNN. [14] Future versions of YOLO have tried to improve on these drawbacks.

YOLO-v3 uses images of dimension, 416x416 for classification, while being trained on ImageNet and the datasets. Images greater than 416x416 are resized, to adapt to the processing time and improve performance, but this degenerates the quality of the image. The use of anchor boxes helps a great deal in the prediction. It complements the recall values, yet negatively affects the total accuracy of the model. This, in turn, lessens the limitations on the model and gives more space to improve. In other words, it can predict more bounding boxes per image, which increases the odds of detection, but at the expense of accuracy. YOLO-v3 struggles with small objects that appear in groups, such as a group of people at a party or large gathering with various lighting situations.

Mask R-CNN, on the other hand, does an excellent job at generating masks of certain classes like Dress and Trousers but fails to distinguish between classes such as Shorts and Skirt, which is an indication of the masks not being generated accurately, for objects that resemble similar shapes. Besides, having more training images with better segmentation ground truth values can help generate better results for categories that have low accuracy scores.

VII. CONCLUSION

In this paper, we presented two techniques for object detection in images. The first generates bounding boxes around the garments using the YOLO-v3 framework. The second uses Mask R-CNN to create masks around the garments in the

images. Our goal was to determine if the background affects the detection of objects within images. To do so, we compare the object detection between the original and cropped images as depicted in Figure 5.

Further, to crop images, we take the output of the bounding box and pass that as an input to the subsequent model to generate masks. We measure the performances of the models created using two metrics, namely Intersection Over Union and Mean Average Precision. From the experiments conducted, we see that we get better results on the original images (40.4%) of the dataset as opposed to the cropped images (34.4%) using mAP. This indicates that the background has a significant influence on object detection and classification in images. To further our claims of the influence of background on image detection, we use Saliency Maps to generate the pixels that were most significant in an image, which were then used to detect objects within images. The Saliency Maps confirm that the background of an image influences the detection of objects within images and cropping an image to omit the background does not improve the accuracy.

An application of our work could be detecting garments in the wild; and classifying the garments using the brand, colour and other parameters, and associating the same with outlets at various geographical locations to improve sales.

In our future research, we are planning to use background subtraction to see if that aids in improving the performance. Background subtraction involves generating the masks and using the detected masks to omit the background entirely and then pass the newly edited image into the model for detection and classification. Additionally, we plan to use YOLO-v5 [11] for bounding boxes. YOLO-v5 is an improvement over YOLO-v3, wherein, with advanced darknet architecture, increased accuracy and less processing time, it generates better bounding boxes around the objects within an image. For instance segmentation we plan to use YOLACT [3] and SOLO [18] as they generate instance masks and calculate the mAP scores using only the masks generated, independent of the bounding boxes.

REFERENCES

[1] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN, 2017.

[2] Shobhit Bhatnagar, Deepanway Ghosal, and Maheshkumar H Kolekar. Classification of fashion article images using convolutional neural networks. In *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pages 1–6. IEEE, 2017.

[3] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 9157–9166, 2019.

[4] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1635–1643, 2015.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Yuying Ge, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5337–5345, 2019.

[7] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *CVPR*, 2019.

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2888–2897, 2019.

[11] Glenn Jocher, Alex Stoken, Jirka Borovec, ChristopherSTAN, Liu Changyu, Adam Hogan, Laughing, NanoCode012, yxNONG, Laurentiu Diaconu, lorenzomammana, wanghaoyang0106, ml5ah, Doug, Jake Poznanski, Lijun Yu , changyu98, Prashant Rai, Russ Ferriday, Trevor Sullivan, Eduard Reñé Claramunt, pritul dave, and yzchen. ultralytics/yolov5: v2.0, July 2020.

[12] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[15] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[17] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[18] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. *arXiv preprint arXiv:1912.04488*, 2019.

[19] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.