Visual Recognition
Group Project 2

Seelam Lalitha(IMT2017027), Ananya Appan(IMT2017004),
Swasti Shreya Mishra(IMT2017043)

May 23, 2020

Git repository link: `https://github.com/SLR1999/VR_Group_Project_2`

1 Problem Statement

A camera is placed at the entrance of a retail shop. Being a company which provides video analytics solutions, we wish to analyze the feed from this CCTV camera. Assuming that people enter one by one at the entrance, frames of the video will be taken at 5 second intervals. Using these snaps, we wish to do the following.

1. Detect the person at the entrance by either creating a bounding box or a mask
2. Determine what kind of dress the person is wearing (Kurti/Formal Shirt/T-shirt/Saree)

2 Approach Taken

2.1 Dataset Creation

In order to create dataset for the entire model (object detection + classification), we picked up "lookbook" videos from youtube where youtubers wear different variations of a single type of attire in a video. We chose around 6-8 videos from each of the categories and generated images out of the video frames to construct the dataset for our classifier model.

The images for the classifier model have been generated separately for Masked RCNN based approach and YOLO based approach by running them individually on each of the videos. The cropped images thus generated were cleaned up manually to remove outliers.

We have used two datasets - **dataset 1 and dataset 2** - for training our classifier. This was because a lot of shirts in dataset 1 were getting classified as t-shirts. We felt that this was a problem with how we had generated the dataset, rather than a problem with our model. Hence, we generated dataset 2, which has only buttoned shirts as t-shirts.

2.1.1 Links to dataset

Link to Masked RCNN dataset:
`https://drive.google.com/drive/folders/1Tu7YI1z1IwRyx48X85CaDzwuW9sZVM7m?usp=`

sharing

Link to YOLO dataset:
https://drive.google.com/drive/folders/1hkNCzl-lUpbGutdHJ5SZEVaCxGnvMJBO?usp=sharing

2.2 Object/Person Detection

- **Mask RCNN Approach:** Here, we predict masks for each person. These masks are obtained from the frames sampled earlier. The steps are as follows :
    1. We first use Mask RCNN pre-trained on Microsoft Coco dataset for object-detection in each frame of the video.
    2. We obtain the mask of the person in the frame, ignoring all other masks obtained.
    3. We then apply this mask on the image and pass it on to the classifier model trained on our generated dataset.
- **YOLO Approach:** Here, we predict bounding boxes along with the attire classification label in real-time. The steps are as follows:
    1. We first use YOLO pre-trained on Microsoft Coco dataset for object-detection in each frame of the video.
    2. If the frame contains a "person" object, we crop the image as per the bounding box predicted by YOLO.
    3. We then pass the cropped image to our classifier model trained on our generated dataset.
    4. The classification label returned by the classifier model is then appended to the text to be annotated on the video's frame.

2.3 Classification

Once we get datasets from Masked RCNN and YOLO, we first divide respective datasets into training and validation sets. We then used fine-tune pre-trained Alexnet, pre-trained Resnet and AppanNet (This was Mediocre Net in our previous assignment) models as per our current attire classification task. We use the model weights trained on the YOLO dataset for real-time classification of attires.

3 Object/Person Detection

3.1 Introduction

Object detection is one of the classical problems in computer vision where you work to recognize what and where – specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn't indicate where the object is located in the image. In addition, classification doesn't work on images containing more than one object.

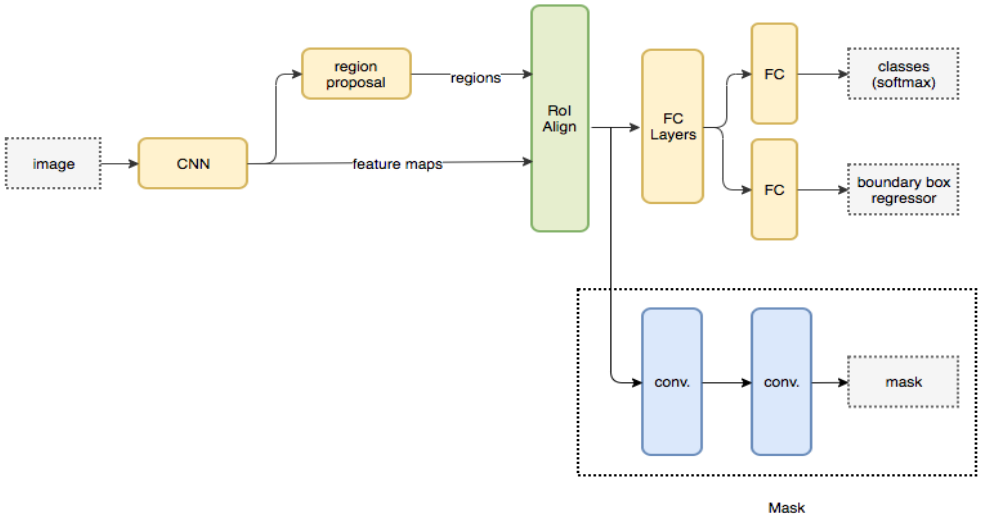
3.2 Masked RCNN

3.2.1 Introduction

Region Proposals are used to find blobby regions which are likely to contain objects. Faster RCNN revolutionized object detection by introducing a Region Proposal Network in order to *learn* the region proposals. Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN.

3.3 Architecture

Shown below is the architecture for Mask R-CNN.



As we can see, after the ROI pooling layer, two CONV Layers have been added in order to generate the mask. This happens in parallel along with bounding box detection.

3.3.1 Accuracies on Masked RCNN

Model	Accuracy
Fine-tuned AlexNet on dataset 1	0.777778
Fine-tuned ResNet on dataset 1	0.680556
Fine-tuned AppanNet on dataset 1	0.597222

3.4 Obtaining the Masks

The following are images before and after applying the mask obtained.

Formal Shirt

T Shirt

As we can see, on obtaining the mask, the entire background is omitted. In some cases, as the one shown above for t shirt, even accessories are omitted in the mask.

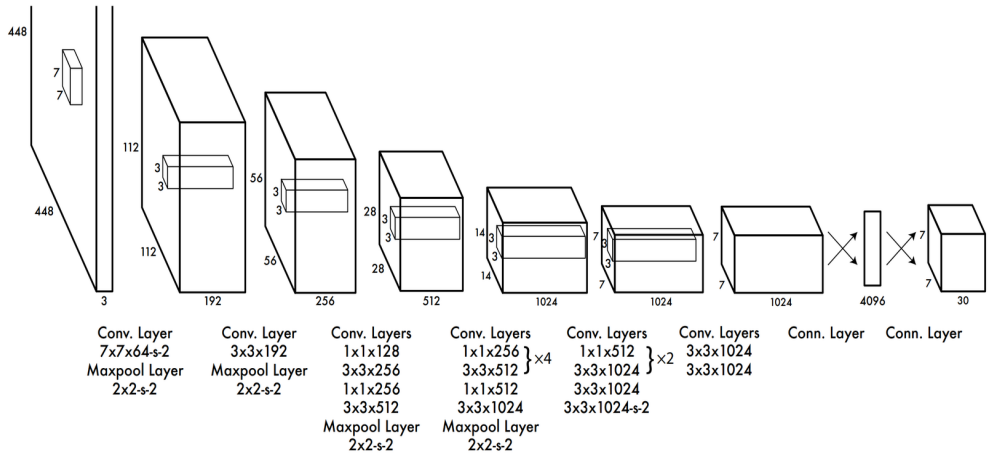
3.5 YOLO

3.5.1 Introduction

YOLO (You Only Look Once) real-time object detection algorithm, has a fully convolutional neural network architecture and passes the image through the network once to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes. The following are the advantages of YOLO:

- YOLO is extremely fast, therefore, it achieves high accuracy while also being able to run in real-time.
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

3.5.2 Architecture



The network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision. The steps taken are as follows:

1. The system divides the input image into an S S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

2. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) IOU . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

3. Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. Each grid cell also predicts C conditional class probabilities, $Pr(Class_i|Object)$. These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

4. At test time we multiply the conditional class probabilities and the individual box confidence predictions, $Pr(Class_i|Object) * Pr(Object) * IOU = Pr(Class_i) * IOU$, which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

3.5.3 Results using YOLO

The cropped images of “person” object obtained after running YOLO over the video stream, are used to train the YOLO classifier model. This model weights are used to get classification

labels to annotate the video. The following are the results obtained after using YOLO and pre-trained Resnet18 classifier on dataset-1 to annotate the video stream:

We can see that YOLO detects both, person and vase with high accuracy.

Our classifier also predicts the attire accurately. Even in the image below, we can see that it has quite accurately distinguished between person wearing saree and person wearing shirt. It has also detected a bottle in the background.

4 Attire Classification

4.1 Fine-tuned AlexNet

We modified the last layer of the AlexNet classifier model to get vector of length 4 and added a softmax layer over it. The rest of the weights of the pre-trained AlexNet model were frozen to avoid their fine-tuning. So, only the last layer along with softmax over it was trained where the index with maximum value gives the prediction label.

4.2 Fine-tuned ResNet

We modified the last fully connected layer of the ResNet18 classifier model to get a vector of length 4 and added a softmax layer over it. The rest of the weights of the pre-trained ResNet model were frozen to avoid their fine-tuning. So, only the last layer along with softmax over it was trained where the index with maximum value gives the prediction label.

4.3 Accuracies on Masked RCNN

Model	**Accuracy**
Fine-tuned AlexNet on dataset 1	0.777778
Fine-tuned ResNet on dataset 1	0.680556
Fine-tuned AppanNet on dataset 1	0.597222

4.4 AppanNet

AppanNet is a medium-deep network. This was Mediocre Net in our previous assignment, and it outperformed a deeper network. We thought we will try to use AppanNet for classification here as well, and compare with ResNet accuracies. This is because deeper networks,

with residual connections, are expected to work better than more shallow ones.

Given below is the code depicting the architecture of AppanNet.

```
    class MediocreNet(nn.Module):

        def __init__(self, activation_fn, zero_padding,dropout_2d, dropout):
            super(MediocreNet, self).__init__()

            self.conv_layer = nn.Sequential(

                # Conv Layer block 1
                nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3,
    padding=zero_padding),
                nn.BatchNorm2d(32),
                activation_fn(),
                nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3,
    padding=zero_padding),
                activation_fn(),
                nn.MaxPool2d(kernel_size=2, stride=2),
                nn.Dropout2d(p=dropout_2d),

                # Conv Layer block 2
                nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3,
    padding=zero_padding),
                nn.BatchNorm2d(128),
                activation_fn(),
                nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3,
    padding=zero_padding),
                activation_fn(),
                nn.MaxPool2d(kernel_size=2, stride=2),
                nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3,
    padding=1),
                activation_fn(),
                nn.MaxPool2d(kernel_size=2, stride=2),
            )


            self.fc_layer = nn.Sequential(
                nn.Dropout(p=dropout),
                nn.Linear(4096, 1024),
                activation_fn(),
                nn.Linear(1024, 512),
                activation_fn(),
                nn.Dropout(p=dropout),
                nn.Linear(512, 10)
            )


        def forward(self, x):
            x = self.conv_layer(x)
            x = x.view(x.size(0), -1)
            x = self.fc_layer(x)

            return x

```

Listing 1: AppanNet

4.5 Accuracies on YOLO dataset

Model	Accuracy
Fine-tuned AlexNet on dataset 1	0.791667
Fine-tuned ResNet on dataset 1	0.833333
Fine-tuned AppanNet on dataset 1	0.652778
Fine-tuned AlexNet on dataset 2	0.861111
Fine-tuned ResNet on dataset 2	0.805556

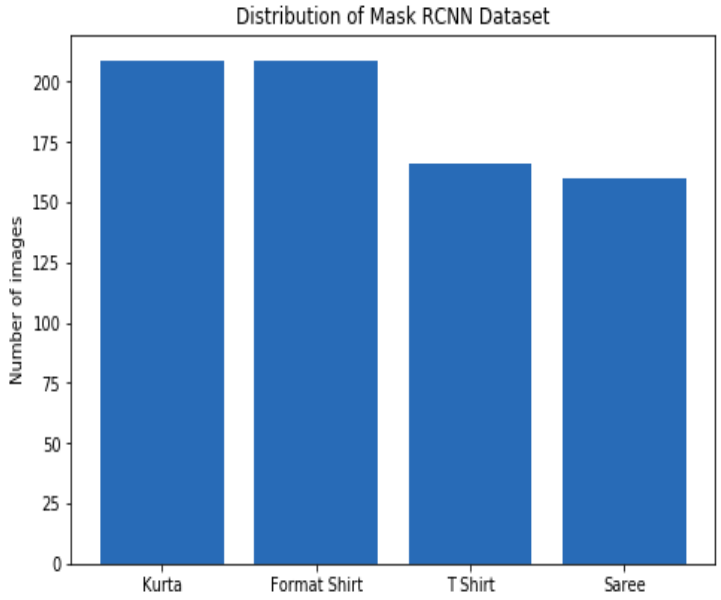
4.6 Results

The following are the results of ResNet classifier on Masked RCNN and YOLO validation datasets respectively.

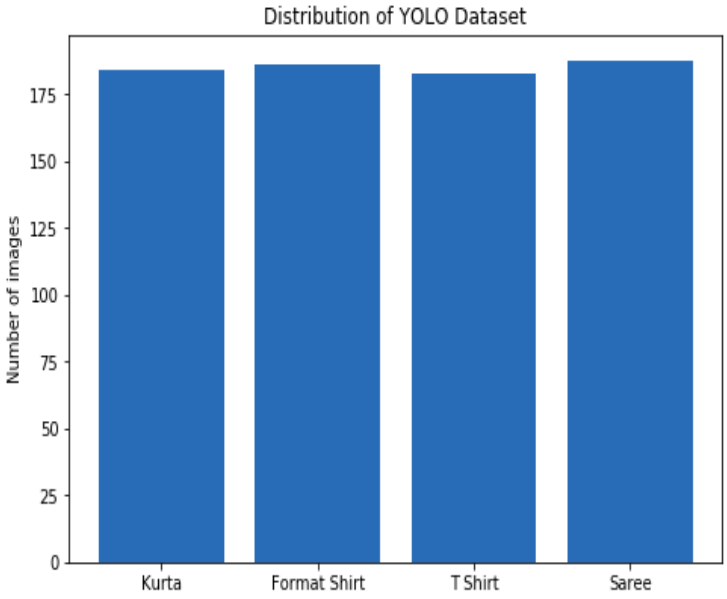
5 Inference

5.1 Distribution of Dataset

The following figure depicts the distribution of data used for Mask RCNN. From the figure it is evident that the number of sarees and t-shirts are less in number compared to the others. So, the dataset is skewed in this. Moreover, as mentioned earlier, in Mask RCNN we tend to lose information as part of image is masked. While accessories are removed from the image, these may actually be important in deciding the category.



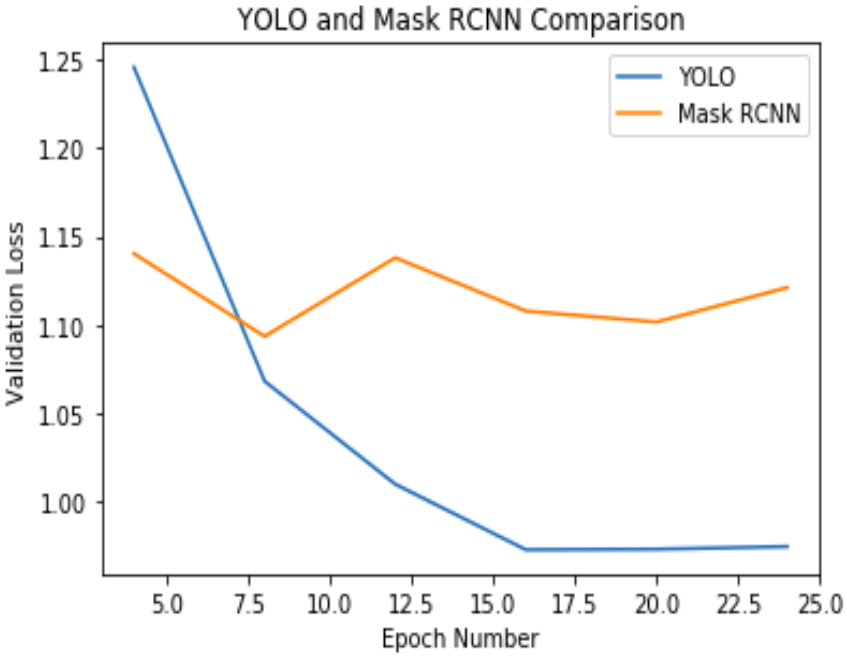
This figure corresponds to data distribution in YOLO. Clearly, different categories are equally present. Also, masking is not involved in this case.



5.2 Validation Loss

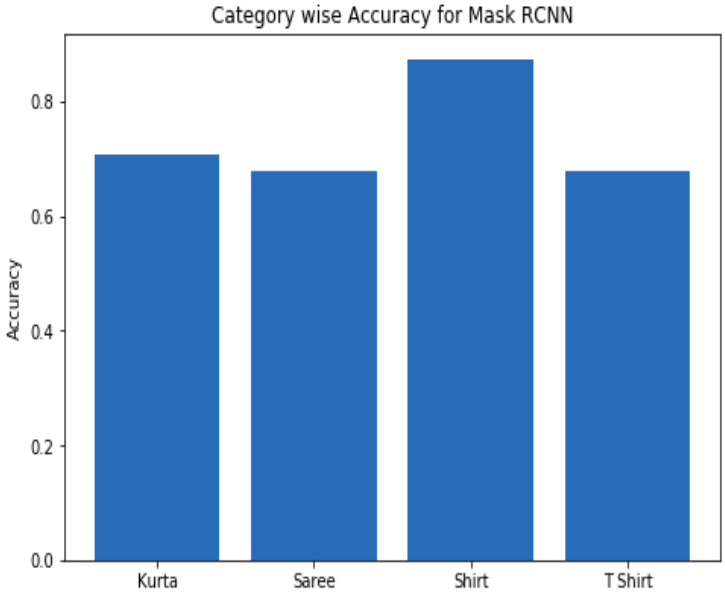
Now, let us try to analyse the following graph which shows the variation of validation loss with respect to epochs in both Mask RCNN and YOLO. From the graph, it is clear that YOLO is learning better. Even though the initialization wasn't good enough in YOLO, it was able to learn at a faster rate whereas eventhough the initialziation was good in Mask

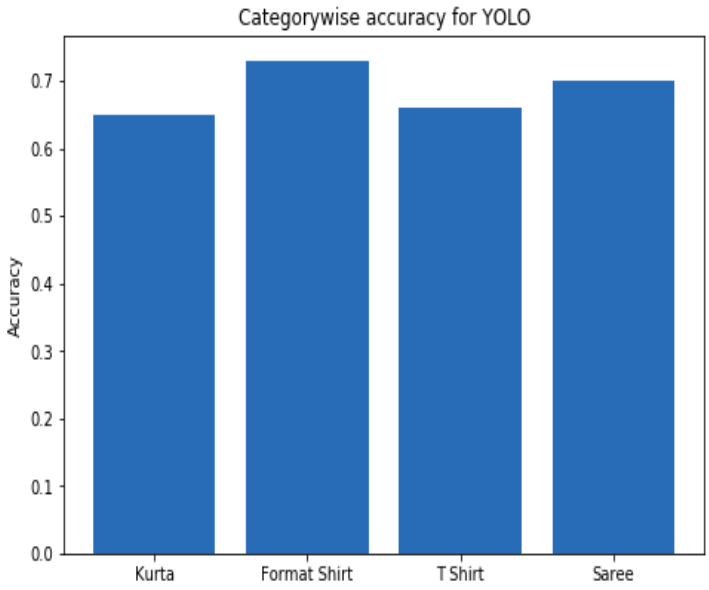
RCNN, it still failed compared to YOLO. The reasons we could come up with are the skewed data and loss of info in Mask RCNN as explained earlier.



5.3 Category-wise Accuracy

Given below are the bar charts depicting the category-wise accuracies for Mask RCNN and YOLO datasets respectively.





We can see that the distribution of category-wise accuracies for Mask RCNN is skewed, while that of YOLO is quite balanced. This is expected, as it reflects the distribution of the collected datasets. We can see that for Mask RCNN, the accuracies of Shirt and Kurta are more when compared to the others. This is because of the nature or the Mask RCNN Dataset, where the number of Kurtas and Shirts is more than T-Shirts and Sarees.

6 References

1. YOLO object detection with OpenCV: `https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/`
2. Overview of the YOLO Object Detection Algorithm: `https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0`
3. YOLOv2: `https://arxiv.org/pdf/1612.08242v1.pdf`
4. YOLO — You only look once, real time object detection explained: `https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006`
5. Mask RCNN Paper : `https://arxiv.org/abs/1703.06870`
6. Mask RCNN Image Segmentation Tutorial : `https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/`