Group Project 2: Dress Recognition

Brahma Kulkarni (IMT2017011)
Kaustubh Nair (IMT2017025)
Sarthak Khoche (IMT2017038)
Vaishnavi Dhulipalla (IMT2017514)

May 27, 2020

1 Introduction

This projects tries to solve the problem of detection of dresses worn by people, given frames from a video feed. We have done this by setting up a deep-learning pipeline that does person detection, followed by dress classification. In this project, we detect and classify the dresses into 4 categories - kurti, saree, formal shirt, and t-shirt. If the model cannot recognize any of these, then a 'Other' label is returned. We have used YOLO for person detection and pre-trained neural networks (Resnet50 and VGG19) for dress recognition.

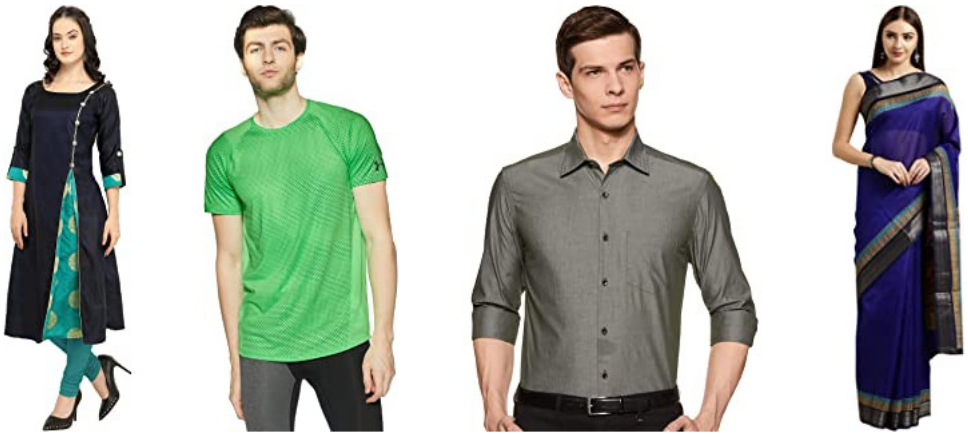

Figure 1: Images of dresses from each of the four classes

2 Dataset curation

The dataset has around 3000 images with approximately 600-800 images for each category of dresses. It was curated by scraping images from various e-commerce websites. We wrote an automatic scraper that downloads the webpage and parses HTML using the *BeautifulSoup*[1] python package. These webpages were downloaded using search keywords such as saree,

kurti, t-shirt, formal shirt, etc. The downloaded images were then sorted manually to remove any unrelated images picked up by the scraper.

3 Person Detection

For person detection we used YOLO which was pretrained on the COCO dataset [2]. YOLO can broadly be used for object detection. Pre-trained with the coco dataset, it can identify 80 different classes of objects. On detecting an object, the algorithm constructs a bounding box around the object. Our focus is more on detecting people. Hence, we restrict the algorithm to only work with people.

4 Dress Recognition

The aim of this section was to classify between different clothing items - T-Shirt, Formal Shirt, Saree and Kurti.

4.1 Dataset

We dataset we used consisted of about 750 to 800 images within each category, scraped from different e-commerce websites like Amazon, FlipKart, Myntra etc. The directory structure is as follows :

```
dataset/
├── 0_formal_tshirt/
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── ...
│   └── 696.jpg
├── 1_kurti/
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── ...
│   └── 929.jpg
├── 2_saree/
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── ...
│   └── 941.jpg
└── 3_shirt/
    ├── 1.jpg
    ├── 2.jpg
    ├── ...
    └── 824.jpg
```

where the first character of each directory is also the class label for that category in the implementation.

4.2 Implementation

In this part, we trained a convolutional neural network from scratch on our dataset. The implementation can be broadly broken down into the following steps :

- Preparing and Reading the Data: Reading images from dataset and passing it through feature extractor to prepare X(array of all images) and y(corresponding class labels).
- Training the Neural Network : The detailed model architecture can be seen below.
- Testing the Neural Network : With a validation split of 0.2
- Saving the model weights : To integrate it with person detector.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 254, 254, 32)      896
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 252, 252, 20)      5780
_________________________________________________________________
flatten_1 (Flatten)          (None, 1270080)           0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               162570368
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 516
=================================================================
Total params: 162,577,560
Trainable params: 162,577,560
Non-trainable params: 0
_________________________________________________________________
None
```

Details of each step can be found in *classification.ipynb*

4.3 Observations and Improvements

With two epochs, the validation accuracy came out around **0.45**. With the resources in hand(Google Colab, which we used to train our neural network from scratch), we added more layers, batch normalization, max pooling, we tried learning rate adjustments, dropout changes and added momentum to see not much of a significant improvement to this accuracy. Hence we switched our focus to **transfer learning** using pretrained neural networks. We primarily used ResNet50 and VGG19, both pretrained on the ImageNet dataset, and observed a significant improvement in accuracy, which is mentioned in the Results section.

We modified each of the pretrained neural network by removing it's last layer and then adding a dense layer with 4 nodes(for our 4 classes) with a *softmax* activation function. Also, while training, all the weights for the pretrained model were directly imported and the

training was done only for the last dense layer!

Details of implementation using ResNet50 can be found in *classification-pretrained-resnet.ipynb* and the details of implementation using VGG19 can be found in *classification-pretrained-vgg19.ipynb*

4.4 Results

With two epochs for training, the validation accuracy for all models came out as :

Model	Accuracy
Our own neural network	0.45
Pretrained Resnet50	0.66
Pretrained VGG19	0.93

5 Integration of Person and Dress Detection

The classification model that was trained and then stored to files [3]. This model was read into the final code files, i.e., yolo.py and yolo_video.py. In these files, first YOLO is used to detect persons, and only when a person is deteced, we draw bounding boxes. So, when a person is detected, the bounding box coordinates are used to crop the frame so that we only need to deal with the portion of the image that has persons.

Having done this, the cropped image is passed through our classification model, and a prediction is made. The prediction output is an array of probabilities for our four classes of dresses. The maximum of these is selected and the index where the maximum occurs tells us about which dress was predicted by our model. Now, if this maximum probability is less than 60%, we are classifying the dress to be, "Other", i.e., it isn't a formal shirt, kurti, saree or t-shirt. The prediction is then printed on the bounding box of the person.

6 Links to Dataset, Code and Classification Model

- Classification Dataset
- Model files
- Code (GitHub Link)

7 Instructions to run the code

- Place the dataset and the model files (.h5 and .json) that you wish to use in the code directory (all links are provided in the next section).
- Place the images you want to test the code on in the images folder.

- Run the following command to test on the provided images. Say the image you want to test on is called 1.jpg, run:
  python3 yolo.py –image images/1.jpg –yolo yolo-coco
- To test on videos, place the videos in the directory called videos. Say the video you want to test on is called 1.mp4, run:
  python3 yolo_video.py –input videos/1.mp4 –output output/1.avi –yolo yolo-coco
- The output video will be written to the output directory.

8 Results

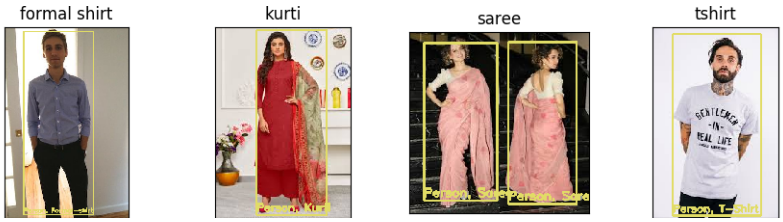

Figure 2: Output

References

[1] Beautiful Soup

[2] YOLO implementation

[3] Saving and loading models