Clothing Recognition

Introduction

Clothing recognition is recently getting more and more attention in the vision community due to its usefulness in real-world applications, such as online social networking, e-commerce, trend analysis, or personal fashion recommender. Recognition of clothing categories is appealing to emerging applications such as security, intelligent customer profile analysis and computer-aided fashion design. This also serves as the basis for more advanced clothing analysis such as parsing, style understanding and attribute recognition.
This project aims to build a clothing recognition system that can be used by a clothing store to recognise the clothes worn by the customers entering the store. The system is shown here to be working on still images but can easily be extended to work on a CCTV surveillance footage. To extend the system to work with surveillance footage, the video needs to be sampled at a fixed sampling rate and fed to the system. It is assumed that the video will be sampled every 5 seconds and at a time only 1 person will be entering the store.

We have tried multiple approaches to the given problem and all the approaches are outlined here.

The dataset can be found here:
https://drive.google.com/drive/folders/1TDOAUYP0qumnfJ4_MF1M0kh4-iFUSB8F?usp=sharing

Solution Outline

The problem can be broken down into 3 sub-problems:

1. **Collection of data for the problem**. This includes collecting images of people wearing clothes of the specified 4 categories – kurti, saree, shirt and t-shirt, and also of people wearing clothes of other arbitrary categories. We need images of the last category so that we can train our model to recognise the 4 specified categories as well as say when the person in the frame is not wearing any of them.

2. **Cleaning the data.** This includes manually going through all the images and removing images that are not appropriate for training the model. This is done so that the model does not learn wrong features from the dataset and can easily generalize to new images.

3. **Detecting a person in the video frame or image**. This includes detecting whether a person is present in the current video frame or image. If yes, we then put a bounding box on the person and crop that much part of the image or video frame. We crop the image so that the cloth recognition model can easily recognize the cloth category as this helps in removing the noisy background. The model can thus learn the features that are most relevant to the problem of cloth recognition.

4. **Recognising the category of the cloth he/she is wearing if a person is detected**. After getting a crop of the person from the video frame or the image, we apply classification techniques for recognizing the category of the cloth the person is wearing.

The approaches tried by us for solving the above 4 sub-problems are described in detail below.

Data Collection

For acquiring our dataset, we tried to locate some established public dataset. Due to unavailability of such a dataset, we evaluated various websites for downloading images so that the data quality was quality and required minimal cleanup. Finally, the data collection was carried out by scraping data from various websites on the internet. We used a Selenium script to do the same which is shown below. The data was scraped from varied sources to maintain diversity in the dataset. We used a mixture of images with plain and non-plain background. The final dataset consisted of about 1300 images per category including the ones from none of the 4 categories.

```
public void downloadImages() throws AWTException, InterruptedException, IOException {
List< WebElement > listOfImages = driver.findElements(By.tagName("img"));
System.out.println("No of images: "+listOfImages.size());
Vector<String> allUrls = new Vector<String>();
System.out.println("Collecting Urls...");
for(int i=0;i<listOfImages.size() && i<maxDownload;i++){
    //System.out.println(i);
    if(!listOfImages.get(i).getAttribute("src").equals("") && !(listOfImages.get(i).getAttribute("src") == null)){
        //System.out.println(listOfImages.get(i).getAttribute("src"));
        allUrls.add(listOfImages.get(i).getAttribute("src"));
    }
}
```

Some of the websites from where the images are taken are listed below:

1. https://www.myntra.com/tshirts
2. https://www.myntra.com/saree
3. https://www.google.com/search?q=kurti&safe=active&sxsrf=ALeKk03rOTKamfYYLqGLxnrD4mpM3DthGA:1589700695149&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiR1O7JsLrpAhUPOisKHd73BucQ_AUoAnoECBAQBA&biw=1366&bih=637
4. https://www.google.com/search?q=kurti&tbm=isch&safe=active&chips=q:kurti,g_1:short:SNoAbrDSm7k%3D&safe=active&hl=en&ved=2ahUKEwjZwI7MsLrpAhUBQ30KHRKMAxYQ4lYoB3oECAEQIg&biw=1351&bih=637
5. https://www.google.com/search?q=saree&tbm=isch&hl=en&safe=active&chips=q:saree,g_1:fancy:_-kQMUJFZyE%3D&safe=active&hl=en&ved=2ahUKEwiDtfKSsLrpAhUBFLcAHY0wB6oQ4lYoCXoECAEQJg&biw=1351&bih=637
6. https://www.google.com/search?q=women+wearing+saree+plain+background&tbm=isch&ved=2ahUKEwjvo7rWr7rpAhW6KrcAHSnfA2YQ2-cCegQIABAA&oq=women+wearing+saree+plain+background&gs_lcp=CgNpbWcQAzoCCAA6BAgAEB46BggAEAUQHjoGCAAQCBAeUKQLWJo0YO81aABwAHgAgAGTAYgBqRGSAQQwLjE3mAEAoAEBqgELZ3dzLXdpei1pbWc&sclient=img&ei=ZefAXq_dBrrV3LUPqb6PsAY&bih=637&biw=1366&safe=active&hl=en
7. https://www.google.com/search?q=people+in+t+shirts&tbm=isch&hl=en&safe=active&chips=q:people+in+t+shirts,g_1:female&safe=active&hl=en&ved=2ahUKEwi14Pyhr7rpAhW7j-YKHbvoCF4Q4lYoD3oECAEQJg&biw=1351&bih=637

8. https://www.google.com/search?q=men+wearing+t+shirt&tbm=isch&ved=2ahUKEwj-lvqMrrrpAhWY5HMBHTihA1YQ2-cCegQIABAA&oq=men+wearing+t&gs_lcp=CgNpbWcQARgBMgIIADICCAAyAggAMgIIADICCAAyAggAMgIIADICCAAyAggAMgIIADoECCMQJzoECAAQQ1DStQJYquUCYJL0AmgAcAB4AIABmAGIAcMNkgEEMC4xM5gBAKABAaoBC2d3cy13aXotaW1n&sclient=img&ei=vuXAXv6_IJjJz7sPuMKOsAU&bih=637&biw=1351&safe=active&hl=en
9. https://www.google.com/search?q=formal+shirt+images+women&tbm=isch&ved=2ahUKEwjStdONrbrpAhULMSsKHcW2CekQ2-cCegQIABAA&oq=formal+shirt+images+women&gs_lcp=CgNpbWcQAzoCCAA6BAgAEB46BggAEAUQHjoGCAAQCBAeUNWxAViP6wFg0-0BaANwAHgAgAGLAogB8AmSAQUwLjguMZgBAKABAaoBC2d3cy13aXotaW1n&sclient=img&ei=s-TAXpKQIovirAHF7abIDg&bih=637&biw=1351&safe=active&hl=en
10. https://www.google.com/search?safe=active&sxsrf=ALeKk00oKGrZMHEG_rbsFlyg7-22Wz1wiw:1589699595181&q=formal+shirt+images&tbm=isch&chips=q:formal+shirt+images,g_1:men:GFlcWWfXUB8%3D&usg=AI4_-kRQpw-dk1H-612LqpD9aDz_7cFjmQ&sa=X&ved=2ahUKEwjc8629rLrpAhVBSX0KHTZIB_UQgIoDKAB6BAgEEAQ&biw=1366&bih=637

Some sample images from the dataset are shown here:






Data Cleaning

Data cleaning is very critical to model performance. Hence, we manually go through all the images and remove the following images:

1. **Duplicate images**: These are removed so that the model doesn't get biased towards features in those images.

2. **Images with multiple persons:** Our project is constrained to allowing only 1 person per video frame. Hence, we remove images with multiple persons.

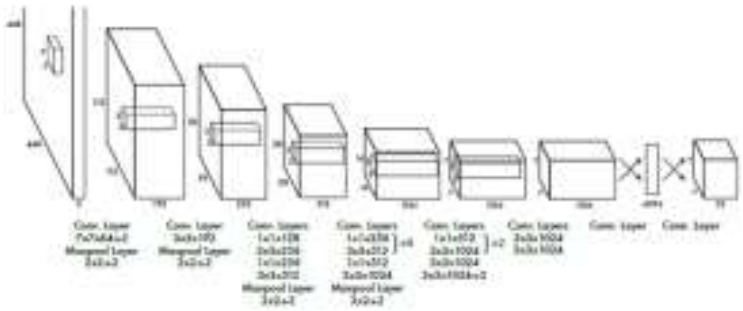
3. **Images having resolution less than 100 x 100:** These are removed as they have very less features for the model to learn.

4. **Images with occlusion:** We remove these too so that the features of cloth categories can be learnt well.

Person Detection

As we know, there might not necessarily be a person in every frame of the video. We need to do cloth recognition in only those images or video frames which capture a person in them. Hence, we first detect whether there is a person in the image and if yes, we crop out a bounding box capturing the person.

YOLO is a CNN based object detection algorithm that applies a single CNN to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. It is currently state-of-the-art object detection model and we use it for person detection.



Faster-RCNN is another object detection model which gives almost same or even better performance sometimes. However, YOLO is much faster than Faster-RCNN and hence, we have preferred to use YOLO over Faster-RCNN. This is advantageous especially in real-time object detection in videos as, we need to finish analysing one frame before the next frame is sampled.

We change the YOLO implementation code to suit our need for person detection and subsequent cropping of the bounding box.

Some sample cropped images are shown below:

Clothing Recognition

After detecting a person in the frame, drawing a bounding box around the person and cropping the image using the bounding box, we give the resultant images to our classification models as input.

We know a convolutional neural network requires a large amount of data so as to learn a large number of parameters and discover the features automatically from the data. Our dataset is not sufficient for training of a convnet since a dataset consisting of a few hundred or thousand examples for each class is considered as small.

Since we only have few examples, our number one concern is **overfitting**. Overfitting happens when a model exposed to too few examples learns patterns that do not generalize to new data, i.e. when the model starts using irrelevant features for making predictions. For instance, if you, as a human, only see three images of people who are lumberjacks, and three, images of people who are sailors, and among them only one lumberjack wears a cap, you might start thinking that wearing a cap is a sign of being a lumberjack as opposed to a sailor. You would then make a pretty lousy lumberjack/sailor classifier.

To overcome the above problem, we have used the following approaches:

1. **Training a small convnet from scratch:**

   Data augmentation is one way to fight overfitting, but it isn't enough since our augmented samples are still highly correlated. The focus for fighting overfitting should be the entropic capacity of your model - how much information your model is allowed to store. A model that can store a lot of information has the potential to be more accurate by leveraging more features, but it is also more at risk to start storing irrelevant features. Meanwhile, a model that can only store a few features will have to focus on the most significant features found in the data, and these are more likely to be truly relevant and to generalize better. There are different ways to modulate entropic capacity. The main one is the choice of the number of parameters in your model, i.e. the number of layers and the size of each layer. Another way is the use of weight regularization, such as L1 or L2 regularization, which consists in forcing model weights to take smaller values.

   In our case we will use a very small convnet with few layers and few filters per layer, alongside dropout. Dropout also helps reduce overfitting, by preventing a layer from seeing twice the exact same pattern, thus acting in a way analogous to data augmentation (you could say that both dropout and data augmentation tend to disrupt random correlations occuring in your data).

   This approach gets us to a validation accuracy of 0.69-0.72 after 50 epochs (a number that was picked arbitrarily --because the model is small and uses aggressive dropout, it does not seem to be overfitting too much by that point).

   Following image shows the above stated model's summary:

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 298, 198, 32)	896
max_pooling2d_3 (MaxPooling2	(None, 149, 99, 32)	0
conv2d_4 (Conv2D)	(None, 147, 97, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 73, 48, 32)	0
conv2d_5 (Conv2D)	(None, 71, 46, 64)	18496
max_pooling2d_5 (MaxPooling2	(None, 35, 23, 64)	0
conv2d_6 (Conv2D)	(None, 33, 21, 64)	36928
max_pooling2d_6 (MaxPooling2	(None, 16, 10, 64)	0
flatten_2 (Flatten)	(None, 10240)	0
dense_3 (Dense)	(None, 128)	1310848
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 5)	645

Total params: 1,377,061
Trainable params: 1,377,061
Non-trainable params: 0

2. **Using Transfer learning:**

   A more refined approach would be to leverage a network pre-trained on a large dataset. Such a network would have already learned features that are useful for most computer vision problems, and leveraging such features would allow us to reach a better accuracy than any method that would only rely on the available data.

   Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

   It has been proven that for 90% or more of the applications, an architecture which currently works best on the ImageNet gives reasonably good performance. So we download a pretrained model and fine-tune it on our data.

   For our application, we used ResNet-50 and VGG-16 for fine-tuning.

   As expected the validation accuracy improved to 91% after 10 epochs. Following images show the summary for VGG-16:

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_6 (Flatten)	(None, 25088)	0
dense_10 (Dense)	(None, 1000)	25089000
leaky_re_lu_5 (LeakyReLU)	(None, 1000)	0
dense_11 (Dense)	(None, 4)	4004

Total params: 39,807,692
Trainable params: 25,093,004
Non-trainable params: 14,714,688

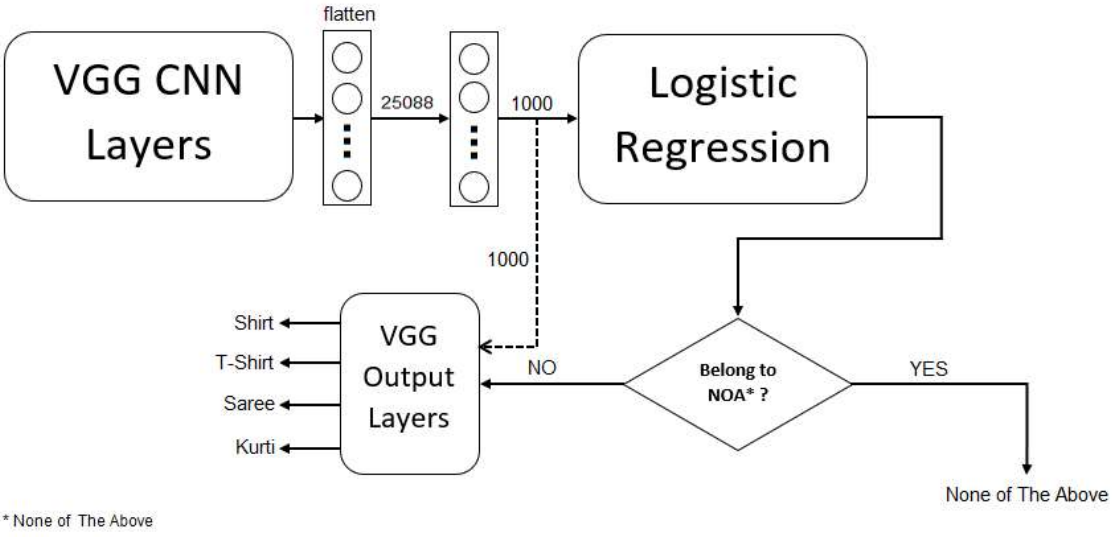
3. VGG-Logistic Regression Hybrid Approach

One of the main challenges in this project was to train the classifier well so that it gives good results in identifying clothes not belonging to any of the 4 classes. So, we came up with yet another architecture to do the same. We thought, since VGG gives really good performance in classifying clothes into the 4 categories, let's train another classifier to classify the images into 2 categories based on whether they belong to none of the 4 classes or in any one of the 4 classes. We then classify the image into one of the 4 categories only if the classifier says that the image belongs to any one category of the 4 cloth categories.

We take a VGG model and add a dense layer at the end with 1000 neurons. We then add another dense layer (output layer) with 4 neurons and softmax activation. We now train this model on train images from the 4 categories. Thus we now have weights of all the layers of the complete model. We now remove the last layer (output layer) and save the model. We now perform prediction on train images from the 4 categories. This gives us feature vectors of length 1000. To prepare this data for being given to Logistic

Regressor, we add a label column and label these images as 1. We now do the same with train images from none of the above category and label them as 0. Both sets are concatenated and given to Logistic Regressor to train it. We then save the Logistic Regressor model. We now get the last layer that we removed along with its weights that we got from training. We add an input layer of 1000 neurons and save the model.

At the prediction time, we first extract the features of the validation set images. We then pass the features to Logistic Regressor. Based on the prediction of Logistic Regressor, one of the 4 things can happen. If the ground truth of the image is none of the above and the Logistic Regressor also predicts the same, then we increment the count of correct predictions by 1. If it had predicted that the image belongs to one of the 4 categories, then we increment the number of incorrect predictions by 1. If the ground truth of the image was that it belongs to one of the 4 categories and Logistic Regressor also predicts the same, then we pass the image features to the last layer model that we had saved along with its actual cloth category. If this model predicts the category correctly then we increment the count of correct predictions and vice versa.



The model architecture of the VGG model is shown below:

```
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_10 (Dense)             (None, 1000)              25089000
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 1000)              0
=================================================================
Total params: 39,803,688
Trainable params: 25,089,000
Non-trainable params: 14,714,688
```

The model architecture of the Logistic Regression model is shown below:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=5000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='saga', tol=0.0001, verbose=0,
                   warm_start=False)
```

The model architecture of the last layer model is shown below:

```
Model: "model_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 1000)              0
_________________________________________________________________
dense_12 (Dense)             (None, 4)                 4004
=================================================================
Total params: 4,004
Trainable params: 4,004
Non-trainable params: 0
```

This approach gave validation accuracy of 78.16%.
The inference time taken by each sample in this approach is about 4.3 seconds without GPU support.

References

We referred to the following sites for developing this project:

1. https://numpy.org/doc/
2. https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
3. https://keras.io/
4. https://pjreddie.com/yolo/