Spring 2018

Outfit Recommender System

Nikita Ramesh
San Jose State University

Outfit Recommender System

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfilment

of the Requirements for the Degree

Master of Computer Science

By

Nikita Ramesh

Spring 2018

SAN JOSÉ STATE UNIVERSITY

The Undersigned Thesis Committee Approves the Thesis Titled

Outfit Recommender System

By

Nikita Ramesh

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

_____

Dr. Teng Moh, Department of Computer Science 5/9/2018

_____

Dr. Katerina Potika, Department of Computer Science 5/9/2018

_____

Prof. James Casaletto, Department of Computer Science 5/9/2018

ABSTRACT

The online apparel retail market size in the United States is worth about seventy-two billion US dollars. Recommendation systems on retail websites generate a lot of this revenue. Thus, improving recommendation systems can increase their revenue. Traditional recommendations for clothes consisted of lexical methods. However, visual-based recommendations have gained popularity over the past few years. This involves processing a multitude of images using different image processing techniques. In order to handle such a vast quantity of images, deep neural networks have been used extensively. With the help of fast Graphics Processing Units, these networks provide results which are extremely accurate, within a small amount of time. However, there are still ways in which recommendations for clothes can be improved. We propose an event-based clothing recommendation system which uses object detection. We train a model to identify nine events/scenarios that a user might attend: White Wedding, Indian Wedding, Conference, Funeral, Red Carpet, Pool Party, Birthday, Graduation and Workout. We train another model to detect clothes out of fifty-three categories of clothes worn at the event. Object detection gives a mAP of 84.01. Nearest neighbors of the clothes detected are recommended to the user.

ACKNOWLEDGEMENT

I would like to express my honest gratitude to my project advisor Dr. Teng Moh, for giving his continuous guidance, precious time and assistance during this project. I am also grateful to my committee members Dr. Katerina Potika and Professor James Casaletto for dedicating their thoughts, time and suggestions.

I would also like to thank San Jose State University and the Computer Science Department for providing me with the required resources for the project.

Last but not the least, I would also like to thank my family and friends for their moral support and encouragement without which I would not be able to finish my project successfully.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Deciding what clothes to wear for an event can often be a time-consuming task. At times, it is important to find clothes that are well-suited for an event. What we wear could have a good or a bad impression on people. Not wearing appropriate clothes on certain occasions can at times offend some people. For example, at a Christian funeral, wearing black conservative clothes is customary while at a Hindu funeral, wearing white conservative clothes is the norm. At Buddhist funerals, wearing the color red is frowned upon. Hence, the problem of event-based clothing needs to be addressed. These days, most of the people share photos of the events they attend on social media platforms. The information obtained from such images could be leveraged to learn the correlation between events and the categories of clothes worn at the events. By learning this correlation, appropriate clothing recommendations could be made.

A recommender system is used to suggest products to customers by using information about the customer, about other customers or about the products and can predict what a customer will prefer [1]. The online apparel retail market size in the United States is worth about seventy-two billion US dollars. Recommender systems generate greater revenue for e-commerce websites if the recommendations are good. Hence, while building such a system for garments, recommendations need to be good.

Object detection is an important part of visual fashion recommendation. Traditional methods for object detection included obtaining feature descriptors like Histogram of Oriented Gradients (HOG), Speeded Up Robust Feature (SURF), Scale Invariant Feature Transform (SIFT), etc. More recently, deep neural networks have been used extensively for object detection. Artificial neural networks are computing systems which are derived from the functioning of the human brain, particularly the nervous system [2]. The main processing units

are neurons, which are connected by links known as synapses. These neurons are divided into various layers. Input data is fed to these neurons in different layers which perform computation on it. There is one input layer and one output layer and multiple hidden layers. If there exists more than one such hidden layer, then the artificial neural network is called a deep neural network. Deep neural networks are much harder to train than normal neural networks. Deep neural networks are used extensively in image processing and could be used to detect objects in images. Using these, we can learn which outfits are worn at which event or used in which scenario. This will help in making recommendations which are tailored to required occasions.

In this paper, we present a novel approach to identify events given an image, and to recognize clothes worn by people in the image using object detection. We make use of Faster RCNN (Region based Convolutional Neural Network), which is an advancement of RCNN [3] and Fast RCNN [4]. Faster RCNN is an algorithm proposed by S. Ren, K. He, R. Girshick and J. Sun [5]. We use this method because this gives us better mean average precision and it also provides faster object detection than its previous versions, as the name suggests. We use transfer learning and create a model according to our needs and train the model on the required data. The model performs well in terms of mean average precision and detects small items very well (Fig 1 (b)). Fig. 1 (a) shows the labeling of an image.

Fig 1 (a): Labeling an image

Fig 1 (b): Clothes detected

The paper is organized as follows: Section 2 presents technologies used for this paper. Section 3 presents the works that are related to this paper. Section 4 explains the datasets used for our work. Section 5 gives a detailed idea about the technical approach proposed. The experiments and results are presented in Section 6. Section 7 then presents the conclusion.

2. Technologies Used

The following technologies were used for the project:

- Tensorflow
- OpenCV
- NumPy
- Protobuf

2.1 Tensorflow

Tensorflow is an open-source machine learning library for high-performance numerical computations. It is now popularly used for computer vision, natural language processing, predictive analytics and many more such applications. Tensorflow, which started off as DistBelief, was developed by the Google Brain team for use at Google and had its first release in 2015. Tensorflow works using directed graphs, where the nodes represent computations and the edges represent tensors. A tensor is an n-dimensional matrix and is the basic unit of computation in Tensorflow.

A big advantage of using Tensorflow is that it can be deployed on not only Graphics Processing Units (GPU) but also Central Processing Units (CPU) and Tensor Processing Units (TPU). It also allows checkpointing of models. That is, we can train a model for a while, stop it, perform evaluation on the model and then start training it back from the checkpoint. Tensorboard also allows visualization of logs of training and evaluation as well as computational graph visualization. Scikit-learn is another Python-based machine learning library. But it is not very useful when it comes to deep learning. For our implementation, we have used Tensorflow Object Detection [31].

2.2 OpenCV

Open Source Computer Vision Library is an open-source library that is used mainly for computer vision and machine learning. It has Python, C++, MATLAB and Java interfaces. It supports multiple operating systems like Windows, Linux, Android, Mac OS, iOS, FreeBSD, Maemo, Blackberry 10, NetBSD and OpenBSD. OpenCV has more than 2,500 algorithms of machine learning and computer vision. OpenCV has multiple applications like facial recognition, object detection, human-computer interaction, finding similar images, ego-motion estimation, etc [6].

For our work, OpenCV was used for making recommendations. We found the images of clothes which were most similar to the clothes suggested by the initial model. OpenCV is well known for this application as it can find similarity between images by calculating the distances in their histograms. A histogram of an image gives us an idea about the intensity distribution of the pixel values. Using OpenCV, if we compare the histograms (calculate histogram distance) of two images, we can find how similar the images are.

2.3 NumPy

NumPy is a Python library for scientific operations. It is used for numerical analysis. It consists of an n-dimensional array object along with other derived objects. It also consists of broadcasting functions, tools for integrating C/C++ and Fortran code, Fourier transform, along with multiple number operations [7]. NumPy arrays or ndarrays form the base of this library. The elements in these arrays are required to be of the same data type. Hence, they will be the same size in memory. Other than its various scientific and numeric applications, NumPy can be utilized as an efficient multi-dimensional package of generic data. NumPy is authorized under the BSD

license, allowing reuse with some restrictions. For purposes of this work, NumPy was used to convert image data into a format which can be used as input for Tensorflow operations.

2.4 Protobuf

Protobuf is the Python Protocol Buffers library. It is a library for serializing structured data. Using this library, one can define how they want their data to be and then can use a generated code to read and write this format of data. It is similar to Extensible Markup Language (XML), but it is speedier and simpler. Google initially developed protocol buffers for internal use but then provided code generators under an open source license. In this work, protocol buffers are used to configure the Tensorflow model and training parameters. Before performing object detection, the protobuf libraries must be compiled.

3. Related Works

A lot of work that has been done in various aspects of clothing recommendations. Works [8]-[18] all deal with recommendations. Earlier works like [9], [17] and [18] made lexical recommendations. Lexical methods like Multimedia Web Ontology Language, Open Mind Common Sense and contextual knowledge have been used in these works in order to make recommendations. All these methods use some form of textual manipulation. In some of the later works [11] - [16], recommendations are visual-based. Images are analyzed instead of textual manipulation, in these works. Recommendations improved in these works because a lot more information is obtained from images.

Earlier works like [9] - [11] find clothes that complement one another. This involves recognition of clothes in the images. Various works like [12], [24] - [26] have worked on clothing recognition. While [24] and [27] deal with image parsing, that is, finding the different components within the image. The works [12] - [14] have worked on clothing recommendations. However, not a lot of research has been done in recommending clothes based on the events (a party, a wedding, a meetup, a red carpet) at which they will be worn. If suggestions consider such events then users will be able to look for clothes specific to their needs and will be inclined to buy the clothes recommended.

The works [10] and [17] make scenario-oriented recommendations. However, [17] uses a text-based methodology. On the other hand, [10] uses Support Vector Machines on images to make recommendations. Of the other works in recommendations, [20] analyzes personal style and [16] deals with clothing analysis based on the location of countries. Personalizing suggestions in this way has been known to increase the chances of a piece of clothing being purchased.

Recommendations can be improved if we analyze and find clothes which people pair together. This could be achieved by using deep neural networks.

Using image processing to make recommendations has gained popularity over the last few years. Neural networks have made a lot of progress in image processing. Neural networks can be trained to identify specific features in images. But when similar features appear at different positions in the image, artificial neural networks cannot identify them [19]. Adding training images for all such images is not feasible. Instead, convolutional neural networks could be used, which identify features without bothering about their position. Works like [11], [15], [19]-[21] use convolutional neural networks to make clothing recommendations. Convolutional neural networks were made particularly popular by Krizhevsky, et al. [21]. They created a network known as AlexNet, which is considered as a turning point for neural networks.

Convolutional Neural Networks (CNN) are a category of neural networks. A CNN usually has a convolutional layer along with a few other layers. The convolutional layer runs by applying the operation of convolution. An image could be represented as a matrix of its pixels. Consider Fig. 2 (a) to be our image matrix. And consider Fig. 2 (b) to be the filter. The convolution operation slides a window of the filter size over the image matrix and computes another matrix which is the convolved feature matrix (Fig. 2 (c)).

0	0	1	1	1
0	0	0	1	1
1	1	1	0	0
1	0	1	1	0
1	1	1	1	0

Fig. 2 (a): Image Matrix

*

1	0	1
0	1	0
1	0	1

Fig. 2 (b): Filter

=

3	2	4
3	3	2
4	4	3

Fig. 2 (c): Convolved Matrix

Recently, Siamese networks and triplet embeddings have also gained popularity [20], [22]. The main benefit of using Siamese networks is that they directly train for the problem at hand. For example, if we are trying to find similarity between two pieces of clothing, then Siamese networks will directly address this issue, instead of training for object detection and then finding the similarity. The research in [20] uses a Siamese CNN to find similar clothes. Triplet embeddings are beneficial because they help improve classification accuracy by extracting better features. The work [22] proposes a bidirectional cross-triplet embedding algorithm. They combine triplet embedding with cross-domain image retrieval.

Scenario/event recognition is an important part of clothing recommendation in the proposed work. In the work [29], event recognition is performed using three steps: event concept discovery, training concept classifiers and prediction of concept scores. Tagged images are used to obtain concepts. Images of correlated concepts are clustered together. A feature vector is formed by taking a concatenation of all concept scores for an image. A classifier then classifies the image as a particular event. The work [30] performs event recognition in photo collections. They use a hidden Markov model for the same. However, this sort of event recognition does not make use of all the information available in the image. It uses tags associated with the image to find out objects within the image. Instead, we can detect objects within the image to identify the type of event.

Using transfer learning has also gained popularity. Transfer learning is the process of using weights of an already trained model in order to train a new model on a similar task. In this way, we can avoid training an entirely new model from scratch and can instead concentrate on improving the task at hand. Transfer learning improves the time required to train a model significantly. In this paper, we make use of transfer learning to train models for object detection. We use Faster RCNN as the meta-architecture for object detection.

RCNN was the idea of Girshick et al. [3]. They came up with a good method to detect objects for the PASCAL VOC challenge, a popular object detection and classification challenge. This method drastically improved the accuracy of object detection as compared to its earlier works. Fig. 3 gives the system overview of RCNN. RCNN works by performing training on three separate models. The first model makes region proposals. These are regions with high possibility of containing an object. Each image has about 2,000 region proposals. RCNN creates region proposals using a method called Selective Search. RCNN will run CNN on each of the region proposals. So, when the region proposals are huge in number, it takes a long time for processing. That is why, RCNN requires about fifty seconds as average testing time per image. The second model trains Support Vector Machines (SVM) to classify which category the object belongs to. After this, the bounding boxes for the model generated are made more precise by training a linear regression model.

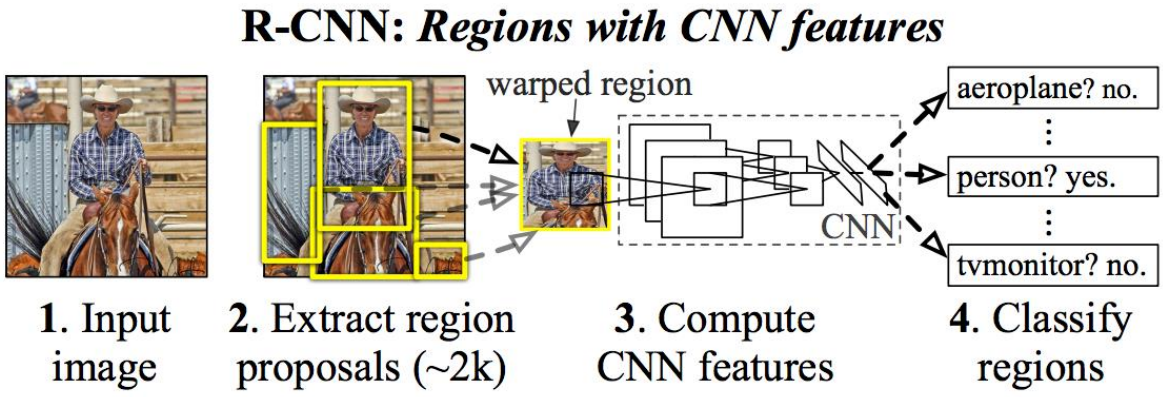


Fig. 3: RCNN System overview [3]

One of the authors of RCNN, R. Girshick, continued working on the challenges faced by RCNN and came up with a new algorithm, Fast RCNN [4]. Fast RCNN solves the issue of the

slow RCNN architecture. In this new architecture, the region proposals are created in a similar manner to RCNN, by using Selective Search. However, CNN is run just once on the entire image instead of on all region proposals. Fast RCNN has a layer called the Region of Interest (RoI) pooling layer. In this layer, CNN features for every region proposed are obtained from the single CNN feature map. Then these features are pooled together. After this, instead of training an SVM model, a softmax layer is added to the model for classification. Also, a linear regression layer is added to the same model to get precise bounding boxes. Thus, just one CNN is run on the entire image. Because of this, Fast RCNN reduces the average testing time per image to around two seconds. Fig. 4 shows the architecture of Fast RCNN.

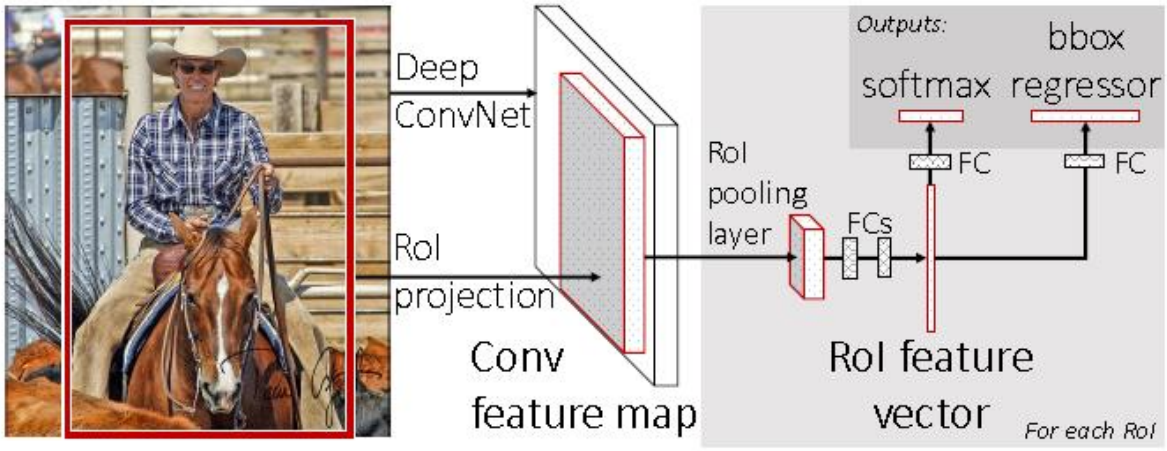


Fig. 4: Fast RCNN architecture [4]

Sometime in 2015, Ren et al. [5] came up with an architecture which was better than Fast RCNN. Even though Fast RCNN was much faster than RCNN, there still was a bottleneck. This was the Selective Search method used for generating region proposals. This step was not necessary as features of the image are calculated when CNN is run. Hence, in Faster RCNN Selective Search is replaced by Region Proposal Network (RPN). RPN generates region proposals. After RPN, the

RoI pooling layer, the classifier and the regressor are similar to the Fast RCNN architecture. Because of this Faster RCNN generates bounding boxes with an average testing time per image of approximately 0.2 seconds. Fig. 5 gives an overview of Faster RCNN.

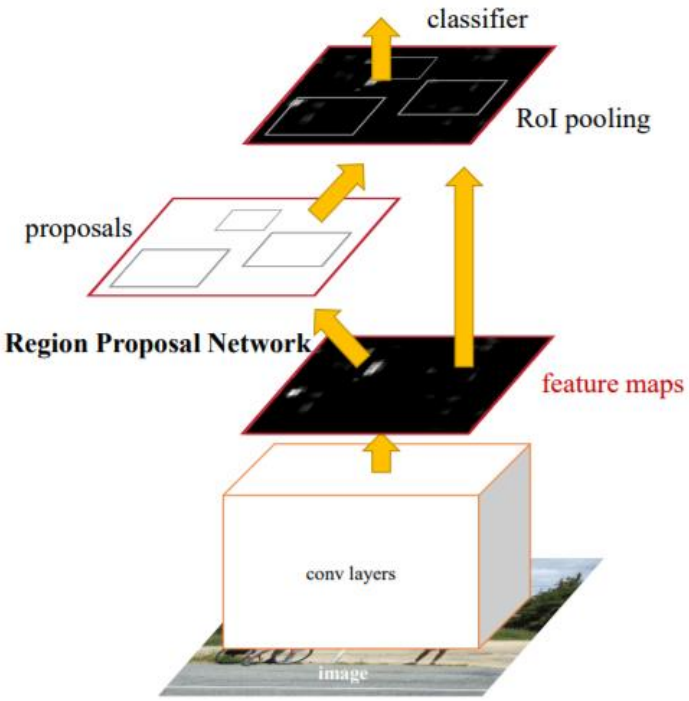


Fig. 5: Overview of Faster RCNN [5]

4. Datasets

4.1 Event Identification

For event identification, we collect data from a variety of sources by crawling the web in addition to the SocEID dataset [29]. The SocEID was created by Ahsan, et al. by querying Instagram and Flickr for event images. We use part of this dataset, according to the events we are considering for this work. We collect 400 images for every event and end up with a dataset of 3,600 images. Along with these images, we also crawl the web for images of objects required to identify the events. We collect 250 images for every object that needs to be detected in these images. We initially experimented with a different number of images starting from 100. The detection was much better with a greater number of images. Hence, we chose 250 images as an appropriate number of images for object detection.

4.2 Clothing Recognition

For clothing recognition, we create a combination of several datasets. These include the Fashionista dataset [33] and the Clothing Co-Parsing (CCP) dataset [25] as well a few online sources. The Fashionista dataset is a collection of 685 images which are fully parsed. This dataset was collected from chictopia.com which is a social networking website for fashion bloggers. This dataset has fifty-three clothing categories like dress, jeans boots, etc. The same fifty-three categories are available in the CCP dataset. The CCP dataset consists of 2,098 images. We collect the remaining clothes belonging to the fifty-three categories of clothes from online sources. In this way, we create a dataset of close to 6,200 images. Fig. 6 shows an example of the images collected.

Fig. 6: Example of the dataset

5. Technical Approach / Methodology

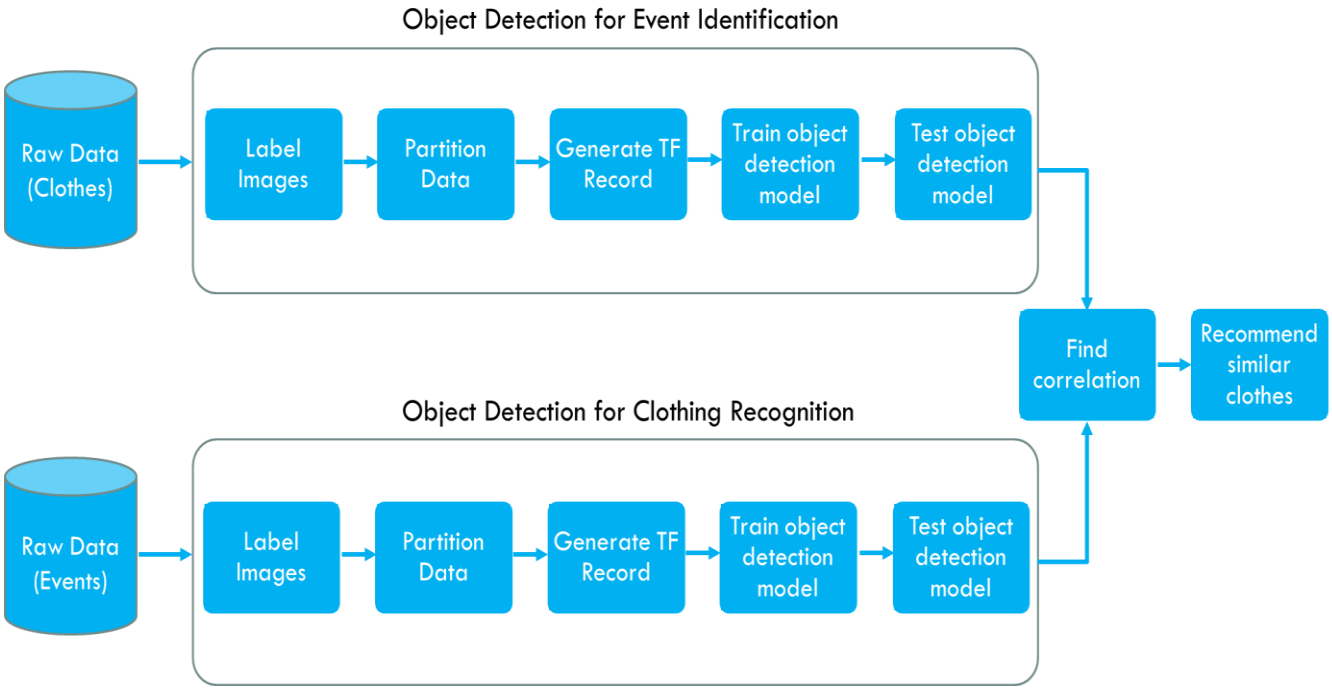


Fig. 7: Pipeline of the Proposed Approach

In this paper, we propose a novel approach for event-based clothing recommendation. We first identify the type of event using object detection. Once we know the event in the image/s, we identify the clothes worn at that event. After this, we find the correlation between the event and clothes worn. We find out the most frequently used clothes and recommend similar clothes using a nearest neighbor approach (section 5.3). Fig. 7 summarizes the pipeline for the proposed approach which is explained in sections 5.1 – 5.4.

5.1 Object Detection for Event Identification

We first collect raw data from a variety of online sources for nine events/scenarios. These events are White Wedding (a semi-formal wedding which is known by that name because the bride wears a white dress), Indian Wedding, Conference, Funeral, Red Carpet, Pool Party, Birthday,

Workout and Graduation. We collected 400 images for every category. To identify the event, we look for objects commonly found at these events. We first need to find which objects are most common at an event. To perform this task, we follow a similar method as that described by Ahsan, et al. [29] to find segments. We get tags related to each event and for the set of events $E = \{e_1, e_2, \ldots, e_9\}$, we have a set of tags $T = \{t_1, t_2, \ldots, t_N\}$ where N is the number of tags collected. The goal is to find commonly occurring objects. We obtain the commonly occurring objects $O = \{o_1, o_2, \ldots, o_n\}$ in each tag. We do this for all event images. We then obtain the final list of commonly occurring objects as follows:

$$argmax_{o_1, o_2, \ldots, o_n} Sc(t_i) = \sum_{j=1}^{n} Sc(o_j) \quad (1)$$

where, Sc is the score of an object. This score is measured by the probability that part of the text is a named entity (in our case, an object). Table I shows an example of the frequently found objects found for a white wedding and at a graduation.

Table I
Example of commonly occurring objects found

White Wedding	Bride, Groom, Flowers, Cake
Graduation	Graduation Cap, Graduation Gown, Degree

Now for event identification, we look for these objects in the images. For this, we train a model to detect these objects. We obtain 250 images for every object and train Tensorflow Object Detection, starting with image labeling. In order to label images, we write a Python script. Using this script, we can drag bounding boxes around objects and label them with their appropriate names. This will generate an XML file with the bounding box information and more information about the image like the file name and file location. This XML file can later be used to feed data into Tensorflow in the required format. We label all the images for the frequently occurring objects. After labeling the data, we partition it into train, test and validation sets. We split the data

into 70% for training, 10% for validation and 20% for testing. We pick this split after a variety of experiments as the best split. For Tensorflow object detection, all labeled training data needs to be in the TFRecord file format. Hence, the labeled training data in XML format needs to be converted to TFRecord format. This was done by converting XML files to Comma Separated Values (CSV) file format and then converting the CSV file into TFRecords.

After obtaining the TFRecords, we can start training the model to detect the objects. For this, Tensorflow Object Detection allows us to train different models using the same codebase. We use transfer learning to train multiple models for object detection. While choosing which model to use, one can experiment with multiple combinations for choosing the meta-architecture, feature extractor and other hyperparameters. For event identification, after experimenting with multiple models, we choose Faster RCNN (section 2) with Inception Resnet v2. After this, we test the model to find out how well it is detecting objects required.

We now use this model to identify the type of event, based on the objects found. For example, if in an image a Graduation Cap and Graduation Gown were detected, then the image is classified as a Graduation event. Fig. 8 shows an example.

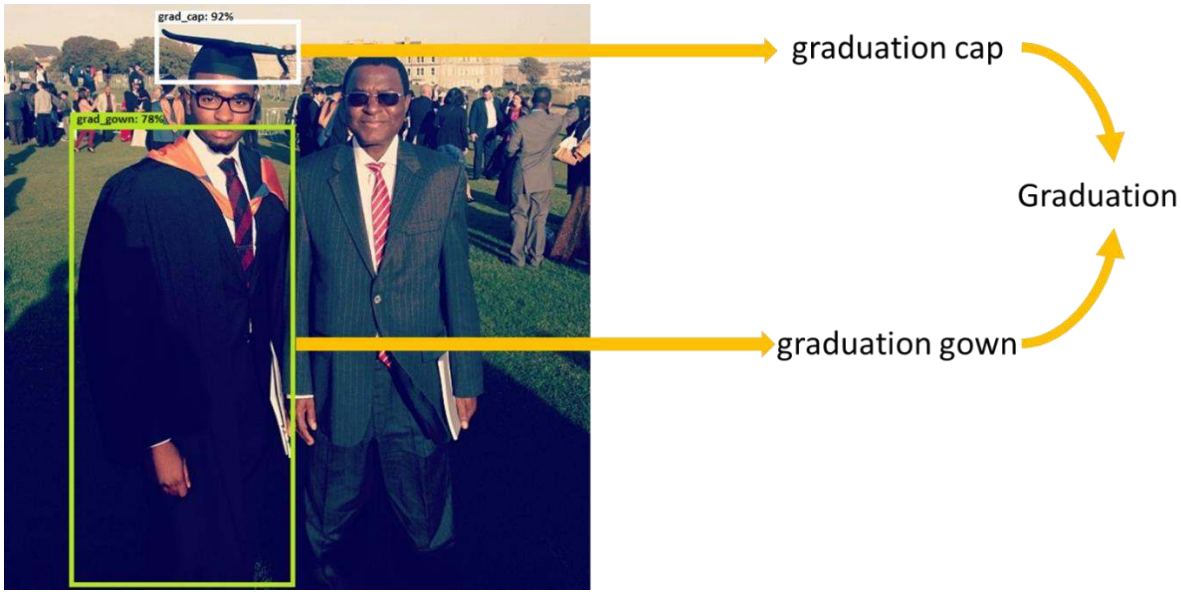


Fig. 8: Event Identification using object detection

5.2 Object Detection for Clothing Recognition

After an event is identified, we need to find out what clothes are worn at the event. For this, we need to train another model to identify clothes. We use a combination of the Fashionista dataset [33], the CCP dataset [25] and online sources for training. Once we have the data, we follow a similar procedure for detecting objects as described in section 5.1. We first begin by labeling all the data we found from the online sources. There are fifty-three categories of clothes in the dataset which we label and obtain the XML files for. We then split the data into training, testing and validation sets. Again, we found that 70% training, 10% validation and 20% testing split is the best split. Then we convert the XML files into CSV and CSV files into TFRecord files. We now start training different models using this data. Fig. 9 shows clothes detected in an image.

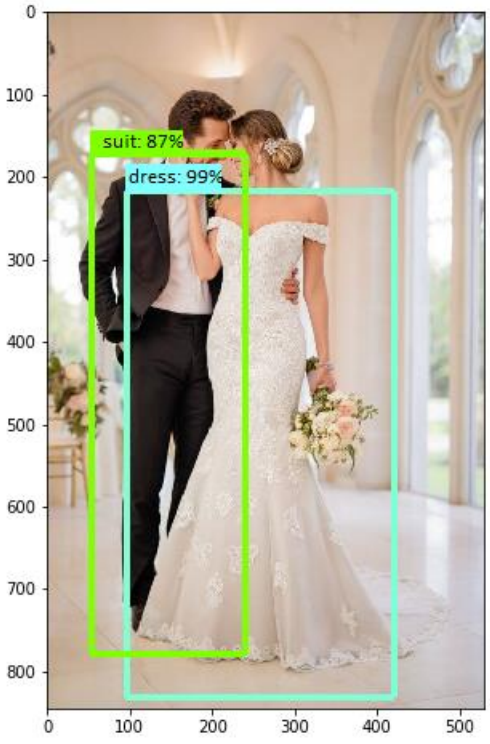


Fig. 9: Object detection for clothes

Tensorflow Object Detection is built on top of Tensorflow. Creating and training new models every time is a time-consuming task. Hence, we have used transfer learning in this paper.

Transfer learning helps in reducing the time required to train models and hence, helps in experimenting with multiple models faster. With transfer learning, we just use the weights from a pre-trained model and apply them according to our need. We can also modify various hyperparameters to find the best model for our data. We can try out from a variety of combinations of architectures to train our model. These are given in Table II [32].

Table II
Object detection models with different meta architectures, feature extractors and other hyperparameters

Meta Architecture	**Feature Extractor**	**Other hyperparameters**
SSD	Inception Resnet V2	Bounding box encoding
Faster-RCNN	Inception V2	Loss functions
R-FCN	Inception V3	Stride
	Resnet 101	Matching
	MobileNet	
	VGG 16	

5.3 Finding Correlation between clothes and events

After we identify the event and detect clothes worn at the events, we find out the correlation between clothes worn and the event. We do this by finding out the clothes worn at all events and store this data in the form of a matrix. If any of the fifty-three items of clothing are detected at an event, then we put in an entry in the matrix. We then find the most frequently worn outfits at these events using Pandas. We retrieve these frequently worn outfits to help us recommend items to users. Table III gives an example of the frequently worn outfits.

Table III
Frequently found outfits

Event	1st	2nd	3rd
Conference	suit	blazer	shirt
Red Carpet	dress	blazer	tie

5.4 Recommend similar clothes

Once we find the top categories of clothes, we obtain items of clothing from the event images. These items of clothing have been worn at the events. The assumption is that if similar clothes are recommended to the users, they will be appropriate. Hence, we find out similar clothes using the nearest neighbor parse approach [24]. We learn a local appearance model for every item of clothing that is to be suggested. Then we find out the nearest neighbors for this item of clothing. Fig. 10 shows results of the similar clothes found for a funeral.

Fig. 10: Similar clothing results for funeral

6. Experiments & Results

6.1 Object Detection

For our experiments we train a variety of object detection models. We pick the following combinations for our experimentation on clothing item detection.

1. SSD (Single Shot Multibox Detector) Mobilenet
2. SSD Inception v2
3. Faster RCNN Resnet 101
4. Faster RCNN Inception Resnet v2

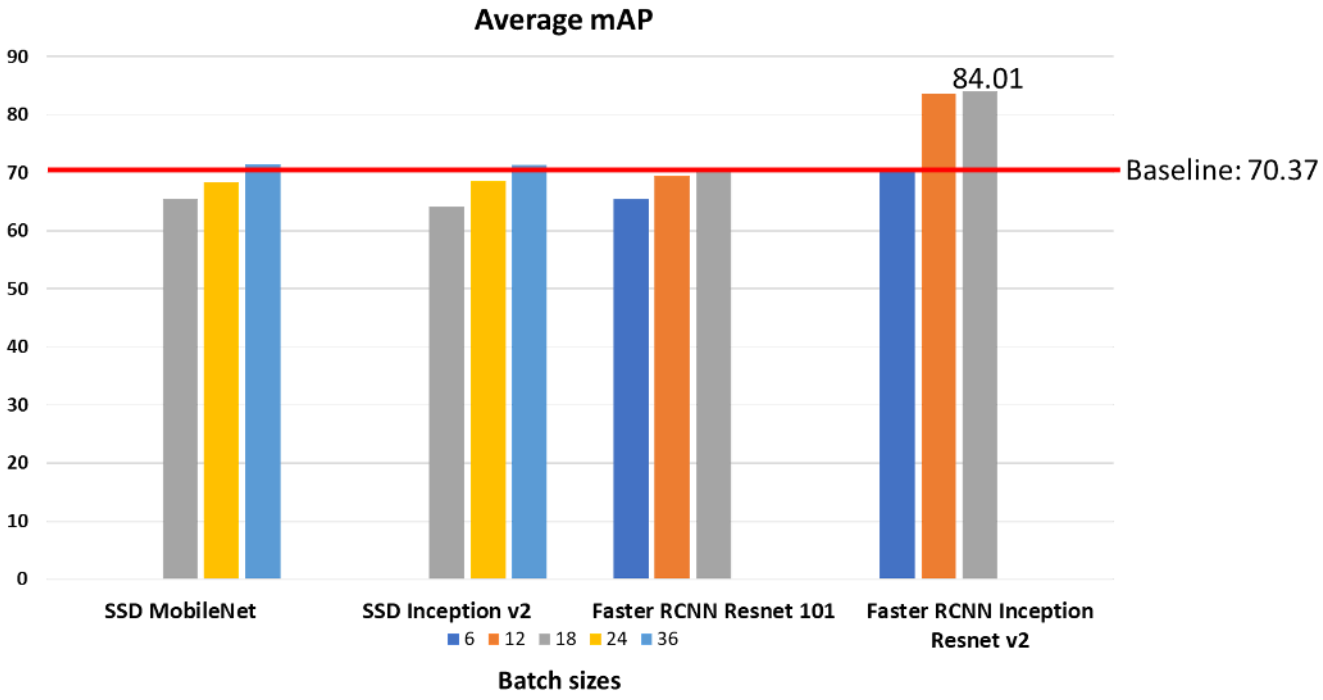


Fig. 11: Average mAPs of different models

Fig. 11 shows the experimented mean average precisions (mAPs) for the four models mentioned above. The legend below shows the batch sizes used for experimentation. The mAP values are compared to the baseline and not to one another. Mean average precision is a popular method to measure the accuracy for object detection. Average precision is calculated as the average of maximum precisions at different recall levels. This is calculated for all classes of objects to be

detected in your model. A mean of these values is called the mean average precision. We get the highest mAP of 84.01 using the Faster RCNN Inception Resnet v2 model.

We started training with SSD along with MobileNet as the feature extractor. This model trains fast as compared to the other models. This is because it performs all tasks in a single forward pass of the network. Also, in SSD models, the images are resized to a fixed shape. Hence, they can work faster on smaller sized images. We keep the stride size 16. We try batch sizes of 18, 24 and 36 and find that the mAP increases with batch size.

Next, we trained SSD with Inception v2 as the feature extractor. This model was also fast because SSD works faster. We keep the stride size 16. Again, we try batch sizes of 18, 24 and 36 and again the mAP increases as the batch size is increased. Thus, we find that the maximum mAP that the SSD models reach with both the feature extractors is roughly around 70. This is because SSD models do not detect small objects accurately. In our fifty-three categories of clothing, we have small objects like shoes, gloves, belt, etc. Hence, we try Faster RCNN models.

Faster RCNN models take longer to train than SSD models. But they generally provide higher accuracy than SSD models. We first trained Faster RCNN with Resnet 101 as the feature extractor. We use a stride size of 16. Because Faster RCNN models train a lot slower, we have to use smaller batch sizes. We try batch sizes of 6, 12 and 18. The mAP is not very different from the SSD models. One of the reasons could be the reduced batch size.

We finally try Faster RCNN with Inception Resnet v2 as the feature extractor. We use a stride size of 16 again. This combination also does not allow larger batch sizes. Hence, we try batch sizes of 6, 12 and 18 again. The mAP for batch size 6 itself crosses 70. Hence, we increase the batch size to 18 and get a final mAP of 84.01. This crosses the baseline clothing item detection [15] by about 13.

6.2 Clothing Recommendation

For clothing recommendation, we first find the frequently worn clothing items at various events. We then try to find similar clothing using the nearest neighbor parse approach. We find the Normalized Discounted Cumulative Gain (NDCG) for the recommendation in a similar manner to the baseline [10]. The Discounted Cumulative Gain (DCG) is found as given in Equation (2).

$$DCG = \sum_{j=1}^{k} \frac{2^{rel(j)}-1}{\log(1+j)} \tag{2}$$

Where, rel is the relevance score of the sample. Using the DCG, we can find the NDCG as given in Equation (3). IDCG is the Ideal Discounted Cumulative Gain. The value of NDCG lies between 0 and 1.

$$NDCG = \frac{DCG}{IDCG} \tag{3}$$

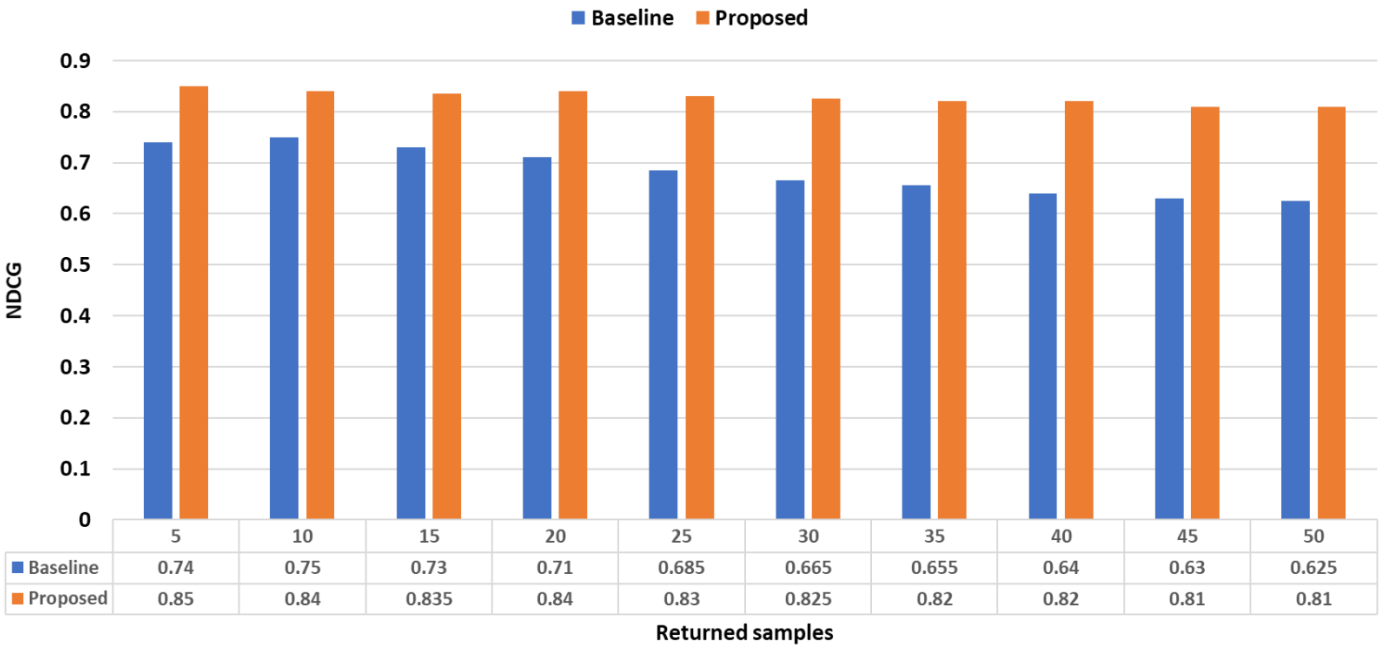
After initial calculations of NDCG values, we find that the NDCG values for a small number of returned samples are high. But the NDCG became smaller as the number of returned samples increased. Hence, we try to improve the accuracy of the samples returned by nearest neighbors. We find that there is a lot of background information that is unnecessary, which might reduce the accuracy of the nearest neighbors found. Hence, we crop the images to reduce the background information. Fig. 12 shows the cropping process.

Fig. 12: Image cropping

This increases the accuracy of the returned samples by quite a bit. We also try to increase the number of data points using some data augmentation techniques. This includes adding background noise and random flipping (Fig. 13).

Fig. 13: Background noise and Random flipping

In our work, we use the nearest neighbors approach as compared to SVM used by the baseline. By cropping, we increase the accuracy of the returned samples and by increasing the data points we make sure nearest neighbors performs well for a greater number of samples. Thus, we find that the NDCG improves. Fig. 14 shows the final results of the NDCG.



	5	10	15	20	25	30	35	40	45	50
Baseline	0.74	0.75	0.73	0.71	0.685	0.665	0.655	0.64	0.63	0.625
Proposed	0.85	0.84	0.835	0.84	0.83	0.825	0.82	0.82	0.81	0.81

Fig. 14: NDCG vs Number of returned samples

7. Conclusion

Recommending clothes using images has made tremendous progress over the years. E-commerce websites are hugely benefitted by this. As research in this field continues, more and more interesting methods have come to light. Work once started using text-based methods, turned to visual methods with image processing and use of neural networks, convolutional neural networks and now transfer learning with deep neural networks.

Thus, we know that there is a common theme in the recent research carried out in the field of clothing recommendation. This theme is analyzing images, finding out features in the images and classifying pieces of clothing in the image. One can understand that this methodology works for most systems. Also, the existing scenario-based recommendations for clothes do not fully utilize the capability of deep neural networks. This paper has introduced a novel approach to recommending clothes based on events and can be used to give better suggestions to its users.

A future work for this paper could be to use the nearest neighbor approach on an online store database instead of the current clothing database to suggest clothes. A user could then directly buy the recommended clothes if he/she wants to.

REFERENCES

[1] F. Isinkaye, Y. Folajimi and B. Ojokoh, "Recommendation systems: principles, methods and evaluation", *Egyptian Informatics J.*, vol. 16, no. 3, pp. 261-273, 2015.

[2] [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network, [Accessed: 12-Apr-2018].

[3] R. Girshick *et al*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition,* 2014. DOI: 10.1109/CVPR.2014.81.

[4] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV),* 2015. DOI: 10.1109/ICCV.2015.169.

[5] S. Ren *et al*, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 39, *(6),* pp. 1137-1149, 2017. DOI: 10.1109/TPAMI.2016.2577031.

[6] [Online]. Available: http://en.wikipedia.org/wiki/OpenCV, http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html, [Accessed: 10-Apr-2018].

[7] [Online]. Available: https://docs.scipy.org/doc/numpy-1.14.0/reference/, [Accessed: 10-Apr-2018].

[8] F. Isinkaye, Y. Folajimi and B. Ojokoh, "Recommendation systems: principles, methods and evaluation", *Egyptian Informatics J.*, vol. 16, no. 3, pp. 261-273, 2015.

[9] D. Goel, S. Chaudhury and H. Ghosh, "Recommendation of complementary garments using ontology", *2015 Fifth Nat. Conf. on Comput. Vision, Pattern Recognition, Image Process. and Graph. (NCVPRIPG)*, 2015.

[10] S. Liu, et al., "Hi, magic closet, tell me what to wear", *Proc. of the 20th ACM Int. Conf. on Multimedia - MM '12*, 2012.

[11] J. McAuley, C. Targett, Q. Shi and A. van den Hengel, "Image-based recommendations on styles and substitutes", *Proc. of the 38th Int. ACM SIGIR Conf. on Res. and Develop. in Inform. Retrieval - SIGIR '15*, 2015.

[12] M. Kiapour, X. Han, S. Lazebnik, A. Berg and T. Berg, "Where to buy it: matching street clothing photos in online shops", *2015 IEEE Int. Conf. on Comput. Vision (ICCV)*, 2015.

[13] V. Jagadeesh, R. Piramuthu, A. Bhardwaj, W. Di and N. Sundaresan, "Large scale visual recommendations from street fashion images", *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining - KDD '14*, 2014.

[14] Y. Hu, X. Yi and L. Davis, "Collaborative Fashion Recommendation: A Functional Tensor Factorization Approach", *Proc. of the 23rd ACM Int. Conf. on Multimedia - MM '15*, 2015.

[15] X. Zhang, et al., "Trip outfits advisor: location-oriented clothing recommendation", *IEEE Trans. on Multimedia, pp. 1-1*, 2017.

[16] E. Simo-Serra, S. Fidler, F. Moreno-Noguer and R. Urtasun, "Neuroaesthetics in fashion: modeling the perception of fashionability", *2015 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, 2015.

[17] E. Shen, H. Lieberman and F. Lam, "What am I gonna wear?: scenario-oriented recommendation", *Proc. of the 12th Int. Conf. on Intelligent user interfaces - IUI '07*, 2007.

[18] L. Yu-Chu, Y. Kawakita, E. Suzuki and H. Ichikawa, "Personalized clothing-recommendation system based on a modified bayesian network", *2012 IEEE/IPSJ 12th Int. Symposium on Applications and the Internet*, 2012.

[19] S. G. Eshwar, G. G. Prabhu J, A. V. Rishikesh, A. N. Charan and V. Umadevi, "Apparel classification using convolutional neural networks", *2016 Int. Conf. on ICT in Business Industry & Government (ICTBIG)*, 2016.

[20] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala and S. Belongie, "Learning visual clothing style with heterogeneous dyadic co-occurrences", *2015 IEEE Int. Conf. on Comput. Vision (ICCV)*, 2015.

[21] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.

[22] S. Jiang, Y. Wu and Y. Fu, "Deep bi-directional cross-triplet embedding for cross-domain clothing retrieval", *Proc. of the 2016 ACM on Multimedia Conf. - MM '16*, 2016.

[23] Z. Liu, P. Luo, S. Qiu, X. Wang and X. Tang, "DeepFashion: powering robust clothes recognition and retrieval with rich annotations", *2016 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, 2016.

[24] K. Yamaguchi, M. Kiapour and T. Berg, "Paper doll parsing: retrieving similar styles to parse clothing items", *2013 IEEE Int. Conf. on Comput. Vision*, 2013.

[25] W. Yang, P. Luo and L. Lin, "Clothing co-parsing by joint image segmentation and labeling", *2014 IEEE Conf. on Comput. Vision and Pattern Recognition*, 2014.

[26] S. Liu, Z. Song, G. Liu, C. Xu, H. Lu and S. Yan, "Street-to-shop: cross-scenario clothing retrieval via parts alignment and auxiliary set", *2012 IEEE Conf. on Comput. Vision and Pattern Recognition*, 2012.

[27] S. Liu, X. Liang, L. Liu, K. Lu, L. Lin, X. Cao and S. Yan, "Fashion parsing with video context", *IEEE Trans. on Multimedia*, vol. 17, no. 8, pp. 1347-1358, 2015.

[28] H. Xiao, K. Rasul and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms", *Arxiv.org*, 2018. [Online]. Available: https://arxiv.org/abs/1708.07747.

[29] U. Ahsan, C. Sun, J. Hays, and I. Essa, “Complex event recognition from images with few training examples”, arXiv preprint arXiv:1701.04769, 2017.

[30] L. Bossard, M. Guillaumin and L. Van, "Event Recognition in Photo Collections with a Stopwatch HMM", *2013 IEEE Int. Conf. on Comput. Vision*, 2013.

[31] TensorFlow, "TensorFlow object detection," GitHub. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection. [Accessed:14-Apr-2018].

[32] J. Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors," Corr, vol. abs/1611.10012, 2016. Available: http://arxiv.org/abs/1611.10012.

[33] K. Yamaguchi *et al*, "Parsing clothing in fashion photographs," in *2012 IEEE Conference on Computer Vision and Pattern Recognition,* 2012, DOI: 10.1109/CVPR.2012.6248101.