**Project Title:** Deep Learning Visual E-Commerce Recommender

**Project Team:** Team Royals

**Team Members:** Wayne Nguyen, Hongcheng Jiang, Fei Wu, and Zhaobin Zhang

**GitHub:** https://github.com/waynenguyen303/deep-learning-visual-eCommerce

1. Project Objectives

   a) Significance

   To improve and refine our initial project objectives, making a full working application was not advised at this time. Instead, an emphasis on the fashion data set should be a priority. In the world of big data, it is only as good as its data sets (garbage in, garbage out). For this increment, we want to focus on collecting staged images of the fashion item and categorize it accordingly. This is to help establish a baseline for accuracy of the implemented machine learning / deep learning algorithms. By using just staged images, we can also get a ball park range of our implemented algorithms time efficiency. Since the end goal is to give recommendations to the user, the application should not exceed a certain time threshold. If it takes too long, this can make the user not use the application. Further on, we would want to incorporate 'live' images of the fashion item.

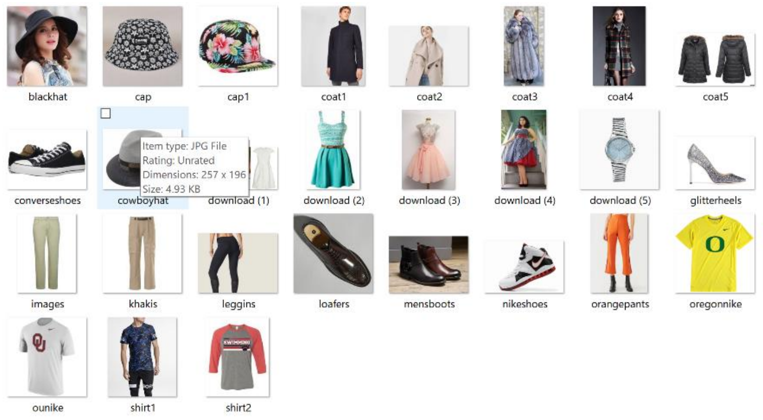
   b) Use Case/Scenario

   A broad use case is to have the user import any image and have our application recommend the fashion item of interest. This of course is a huge problem to tackle. A better starting scenario is to test staged or cataloged images. The fashion item image should be centered, in high definition, and formatted to a certain size. There should be no to little background noise in the image. With this, we would run a three-stage classifier module on the item. The first stage is a deep learning algorithm to classify what the item is. Second, a machine learning algorithm (Clarifai) to classify its design elements and features. The third stage would be a logo recognizer algorithm to see if there is any logos attached to the fashion item. Once the module is done, the image is now correctly tagged with what the item is (watch), what details it has (silver with diamonds), and what brand (Rolex). This tag can then be run through a knowledge recommendation algorithm to show the user where to buy that or a similar item.

2. Data Sets

   a) Initial data set categories.

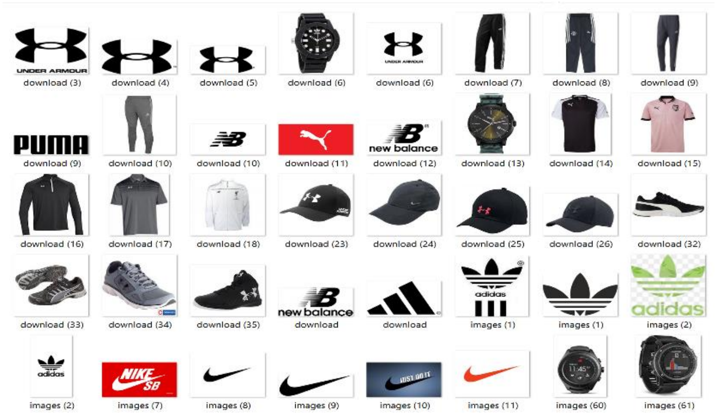
   When first starting this project, the initial intention was to finely categorize the apparel that we would search for online. This could be shoes, coats, T-shirt, pants, dresses, polos, etc. With this notion, we made 9 generic categories for online apparel: dresses, watches, shoes, pants, coats, shirts, suits, Adidas, and hats. The images (.jpg or .png format) were mostly found with Google searches and filed according to their categories. The images were intended to have a lot of diversity, so some had models in them, some with background (live images). Some images had more than one category, but the main subject was of that category. With this, each category had around 100 images each. Of course, we recommend having thousands of images per category for better training but without a proper company/website to facilitate these images, our data set tends to be rather low for each category. The data set was then broken down into training and testing sets. The training set with about 90% of the images per category and the testing set with 10%. Validation images consisted of about 1% of the total images.



As we will see further on, this type of data set was not ideal for image classification or object detection. With how some of the algorithms work, some of the categories were too similar like T-shirts, coats, and polos. Also background and multi-category images tend to make the data set very noisy.

b) Improved data set categories

With this new knowledge of how the classification and object detection algorithm works. Our data set was further refined to get better accuracies. First, we limited the images to catalog style photos. This is where there is very little to no background. No models are included and only the category was in that image, no mixed categories. Also, categories were refined to be only sports related and their logos. So our new category are: sport tops, sport bottoms, sport shoes, sport hats, and watches. Logo categories are: Nike, Adidas, Puma, Under Armor, and New Balance. These categories were chosen to maximize their differences in shape and to hopefully improve accuracies of the implemented classification and object detection algorithms. Again, each category has around 100 images each. Around 90% of the images in each category were for training and 10% for testing.

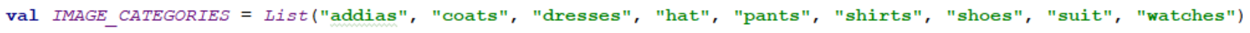


3. Classification and Object Detection Algorithms

## A) Shallow Learning Classification

To do our shallow learning image classification, we will be using the Adobe Spark library using the IntelliJ IDE. We will compare the Naïve Bayes, decision tree, and random forest algorithms on our initial data set and see their accuracies. Given in class, the image classification application from tutorial 4 was modified to run our data sets and the above algorithms. The application can be downloaded here at https://umkc.app.box.com/s/xdf9p87cuuwxxy3ajt3gzog2mcbe3l50. To run this application, the test images needs to be resized and modification to the code needs to be done for each algorithm.

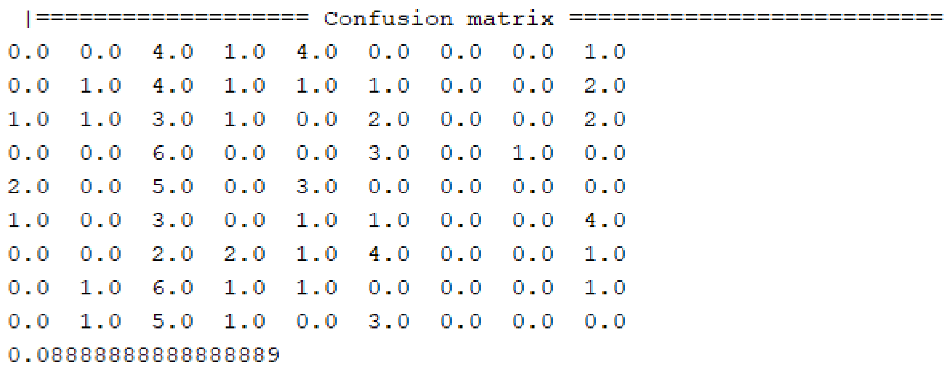
On our initial data set:



```
val IMAGE_CATEGORIES = List("addias", "coats", "dresses", "hat", "pants", "shirts", "shoes", "suit", "watches")
```

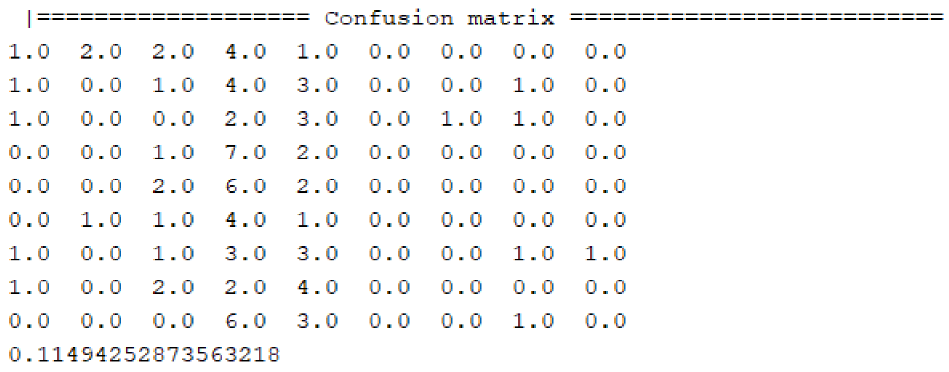
- Accuracy

1) Random Forest Tree



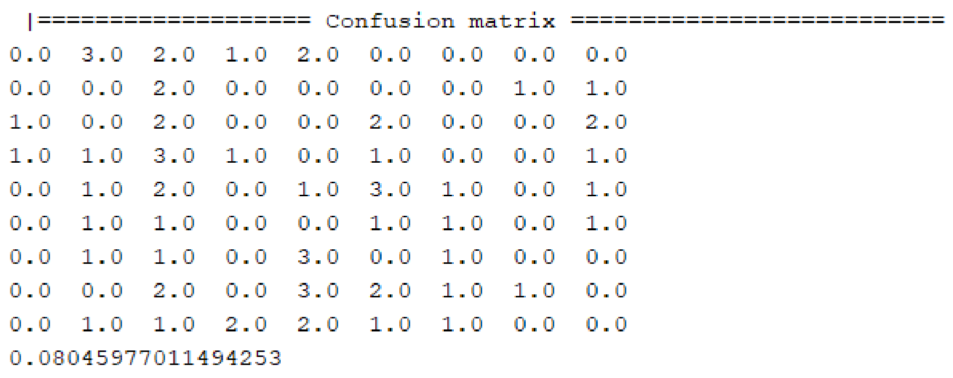
```
 |=================== Confusion matrix ==========================
0.0  0.0  4.0  1.0  4.0  0.0  0.0  0.0  1.0
0.0  1.0  4.0  1.0  1.0  1.0  0.0  0.0  2.0
1.0  1.0  3.0  1.0  0.0  2.0  0.0  0.0  2.0
0.0  0.0  6.0  0.0  0.0  3.0  0.0  1.0  0.0
2.0  0.0  5.0  0.0  3.0  0.0  0.0  0.0  0.0
1.0  0.0  3.0  0.0  1.0  1.0  0.0  0.0  4.0
0.0  0.0  2.0  2.0  1.0  4.0  0.0  0.0  1.0
0.0  1.0  6.0  1.0  1.0  0.0  0.0  0.0  1.0
0.0  1.0  5.0  1.0  0.0  3.0  0.0  0.0  0.0
0.08888888888888889
```

2) Decision Tree



```
 |=================== Confusion matrix ==========================
1.0  2.0  2.0  4.0  1.0  0.0  0.0  0.0  0.0
1.0  0.0  1.0  4.0  3.0  0.0  0.0  1.0  0.0
1.0  0.0  0.0  2.0  3.0  0.0  1.0  1.0  0.0
0.0  0.0  1.0  7.0  2.0  0.0  0.0  0.0  0.0
0.0  0.0  2.0  6.0  2.0  0.0  0.0  0.0  0.0
0.0  1.0  1.0  4.0  1.0  0.0  0.0  0.0  0.0
1.0  0.0  1.0  3.0  3.0  0.0  0.0  1.0  1.0
1.0  0.0  2.0  2.0  4.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  6.0  3.0  0.0  0.0  1.0  0.0
0.11494252873563218
```

3) Naïve Bayes



```
|================== Confusion matrix ========================
0.0  3.0  2.0  1.0  2.0  0.0  0.0  0.0  0.0
0.0  0.0  2.0  0.0  0.0  0.0  0.0  1.0  1.0
1.0  0.0  2.0  0.0  0.0  2.0  0.0  0.0  2.0
1.0  1.0  3.0  1.0  0.0  1.0  0.0  0.0  1.0
0.0  1.0  2.0  0.0  1.0  3.0  1.0  0.0  1.0
0.0  1.0  1.0  0.0  0.0  1.0  1.0  0.0  1.0
0.0  1.0  1.0  0.0  3.0  0.0  1.0  0.0  0.0
0.0  0.0  2.0  0.0  3.0  2.0  1.0  1.0  0.0
0.0  1.0  1.0  2.0  2.0  1.0  1.0  0.0  0.0
0.08045977011494253
```

Each column in the confusion matrix represents a category in the "IMAGE_CATEGORY" list. Accuracies for the 3 shallow learning algorithms on this data set was very low. The random forest algorithm was at 8.88%, decision tree algorithm at 11.49%, and Naïve Bayes algorithm at 8.04%. This is mostly due to the low number of training data images and also the diversity of the images. With this though, the best shallow learning algorithm for this data set is the decision tree. Still these low accuracy scores means there needs to be refinement in the code or data set. We are leaning towards the later.

On our new data set:

**val** *IMAGE_CATEGORIES* = *List*( "bottom", "hat", "shoe", "top", "watch")

- Accuracy

4) Random Forest Tree

```
|================== Confusion matrix ========================
6.0  3.0  2.0  0.0  0.0
1.0  3.0  3.0  3.0  0.0
0.0  2.0  8.0  0.0  0.0
3.0  1.0  2.0  4.0  0.0
0.0  0.0  5.0  2.0  3.0
   0.47058823529411764
```

5) Naïve Bayes

```
    |================== Confusion matrix ========================

7.0  0.0  4.0   0.0  0.0
0.0  1.0  8.0   1.0  0.0
0.0  0.0  10.0  0.0  0.0
1.0  0.0  4.0   5.0  0.0
0.0  0.0  10.0  0.0  0.0
    0.45098039215686275
```

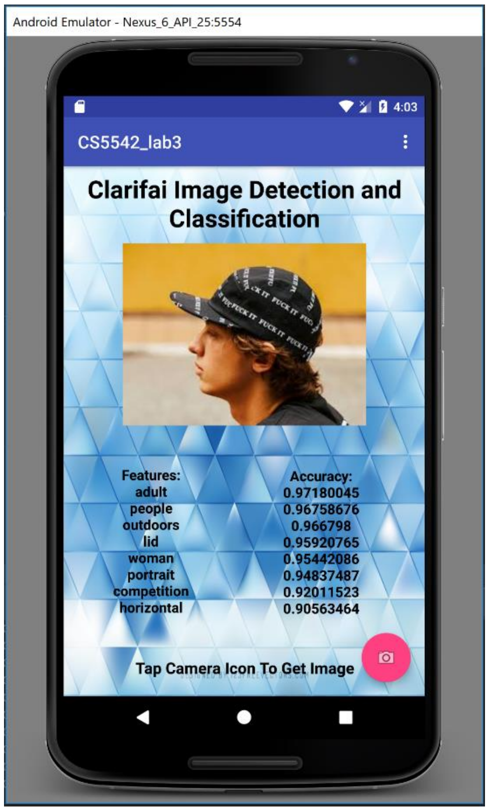
5) Decision Tree

```
|================== Confusion matrix ==========================
6.0  2.0  0.0  3.0  0.0
2.0  3.0  3.0  2.0  0.0
0.0  7.0  3.0  0.0  0.0
1.0  5.0  0.0  4.0  0.0
0.0  5.0  2.0  0.0  3.0
     0.37254901960784315
```

For the new data set, the random forest algorithm its accuracy is 47.05%. The Naïve Bayes accuracy is 45.09% and decision tree at 37.25%. This is much better than the old data set and with more training images, we believe we can get the accuracy to go up even more. Still, we would like to be in the 80% accuracy level before we want to implement this kind of shallow learning algorithm.

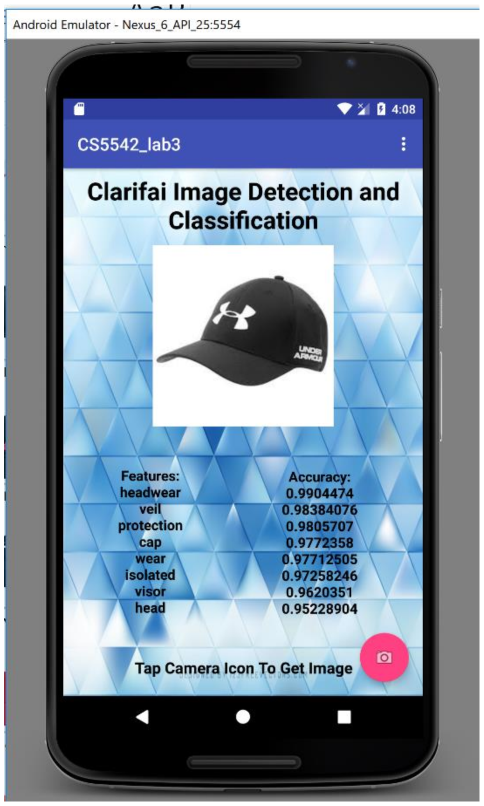
**B) Clarifai Machine Learning for Object Detection**

For object detection, we will use the Clarifai API and its machine learning algorithm. This can be implemented on a web page or on the Android Studio IDE. A Clarifai API key is needed to run its libraries and can be issued at https://www.clarifai.com/. To use Clarifai, an image is sent to its server and a Json response is sent back. The Json has the object detected in the image and also its accuracy.


Android Emulator - Nexus_6_API_25:5554
4:03
CS5542_lab3
Tap Camera Icon To Get Image

Clarifai Image Detection and Classification

| Features: | Accuracy: |
|---|---|
| adult | 0.97180045 |
| people | 0.96758676 |
| outdoors | 0.966798 |
| lid | 0.95920765 |
| woman | 0.95442086 |
| portrait | 0.94837487 |
| competition | 0.92011523 |
| horizontal | 0.90563464 |


Android Emulator - Nexus_6_API_25:5554
4:08
CS5542_lab3
Tap Camera Icon To Get Image

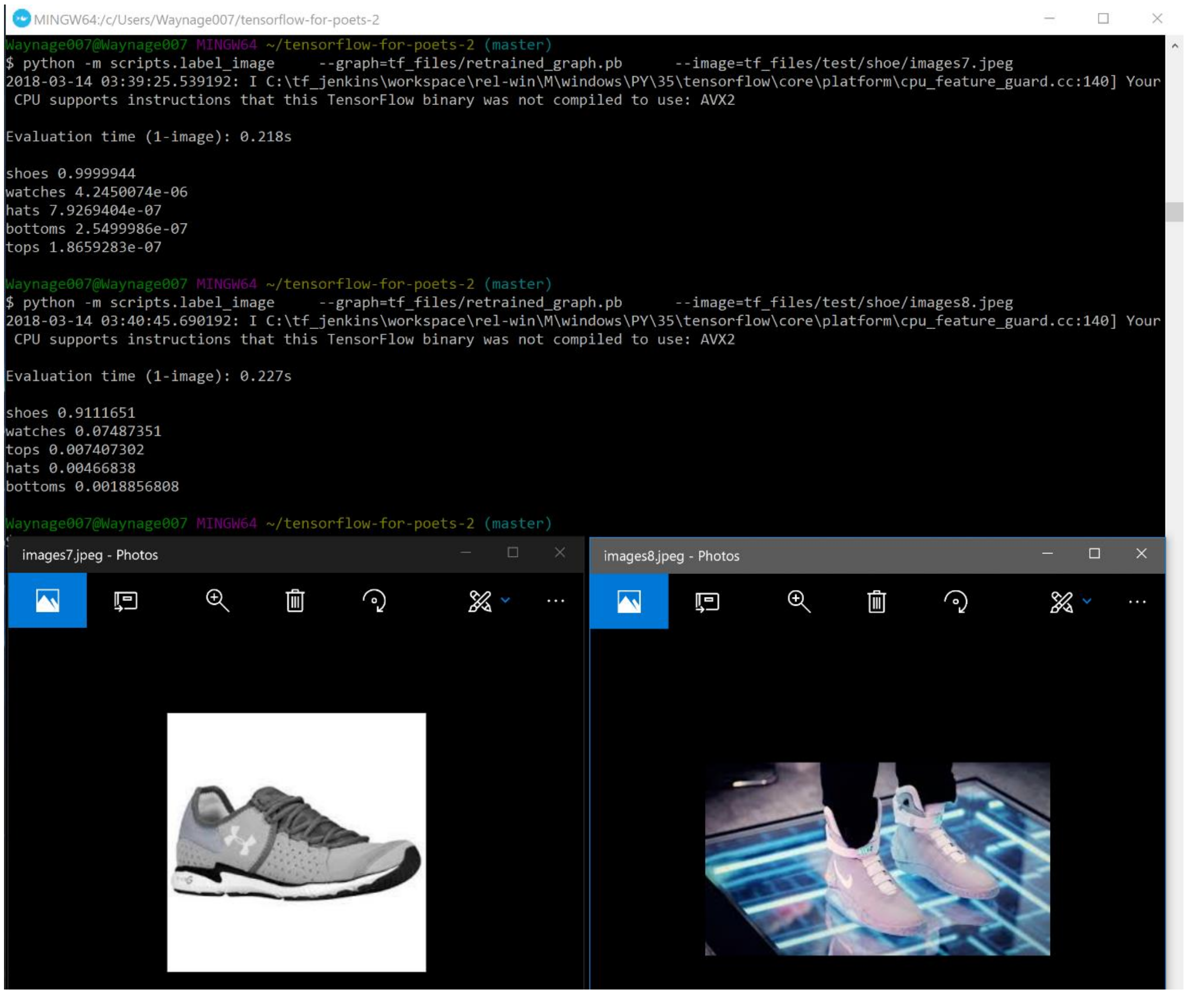
Clarifai Image Detection and Classification

| Features: | Accuracy: |
|---|---|
| headwear | 0.9904474 |
| veil | 0.98384076 |
| protection | 0.9805707 |
| cap | 0.9772358 |
| wear | 0.97712505 |
| isolated | 0.97258246 |
| visor | 0.9620351 |
| head | 0.95228904 |

From the images above, we can see the Clarifai object detection algorithm in action. Its algorithm is accurate and its response time is fast compared to the shallow learning algorithms. What makes a difference in these two is what type of image is sent to Clarifai. For the picture on the left, Clarifai detected ‘adult’, ’people’, ‘outdoors’, and then ‘lid. For the picture on the right, ‘headwear’, ‘veil’, ‘protection’, and ‘cap’ was the top detected words. For our project, we would want the feature names on the right picture. Again this is due to the image selected and sent to Clarifai. This is just another case for the correct data set to be used.

## C) TensorFlow - Deep Learning

For another take at image classification and get better accuracies, we will use TensorFlow and its deep learning algorithm. We will mostly look at the SoftMax implementation. The CNN implementation will be done at a further date. Here we will be using Docker and tensorflow-for-poets to train and test the data set. A tutorial can be seen here at https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/?utm_campaign=chrome_series_machinelearning_063016&utm_source=gdev&utm_medium=yt-desc#0 to implement TensorFlow.



```
Waynage007@Waynage007 MINGW64 ~/tensorflow-for-poets-2 (master)
$ python -m scripts.label_image     --graph=tf_files/retrained_graph.pb      --image=tf_files/test/shoe/images7.jpeg
2018-03-14 03:39:25.539192: I C:\tf_jenkins\workspace\rel-win\M\windows\PY\35\tensorflow\core\platform\cpu_feature_guard.cc:140] Your
 CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

Evaluation time (1-image): 0.218s

shoes 0.9999944
watches 4.2450074e-06
hats 7.9269404e-07
bottoms 2.5499986e-07
tops 1.8659283e-07

Waynage007@Waynage007 MINGW64 ~/tensorflow-for-poets-2 (master)
$ python -m scripts.label_image     --graph=tf_files/retrained_graph.pb      --image=tf_files/test/shoe/images8.jpeg
2018-03-14 03:40:45.690192: I C:\tf_jenkins\workspace\rel-win\M\windows\PY\35\tensorflow\core\platform\cpu_feature_guard.cc:140] Your
 CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

Evaluation time (1-image): 0.227s

shoes 0.9111651
watches 0.07487351
tops 0.007407302
hats 0.00466838
bottoms 0.0018856808

Waynage007@Waynage007 MINGW64 ~/tensorflow-for-poets-2 (master)
```

Here we have trained the categories for: bottoms, hats, shoes, tops, and watches. When testing the images, the accuracy is very good. TensorFlow predicted the image on the left as a shoe at 99.99% and for the image on the right, it predicted it as a shoe with 91.11% accuracy. This is very good compared to the shallow learning algorithms. Also the training time was considerably less using TensorFlow versus the 3 shallow learning algorithms. We will probably be using this algorithm for object classification in our project.

With all this, we have learned that the data set to train and test plays a very important role in how accurate algorithms can be. The less noise the image has and the about of training data is vital for accuracies. This means that more attention to getting the correct data set needs to be the first priority. Second then should be what kind of algorithm to use. For image classification, we would use TensorFlow's deep learning algorithm versus the 3 shallow learning ones. Clarifai's machine learning is good for object detection but we still have to be careful of what images to send it. Maybe a mix of the shallow learning algorithms and Clarifai can improve the list of object detected or wanted.

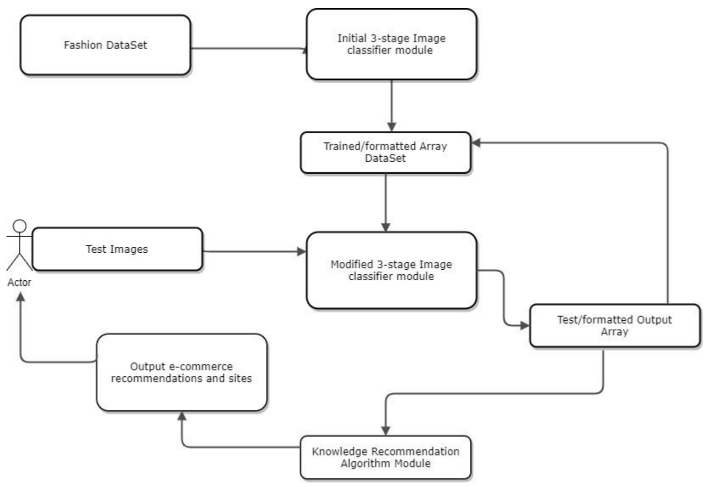
## 4. Related Works

TensorFlow-for-poets : https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/
Docker: https://www.docker.com/
Clarifai: https://www.clarifai.com/
Given shallow learning algorithms (class spreadsheet):
https://docs.google.com/spreadsheets/d/1TkZalFcjQ8MCWLdkQkTdEhUYG5V8hsUF4KzMNmy_Rto/edit?ts=5a5e212f#gid=742202525

Deep learning [3] has become a promising technique to perform various tasks, especially in image/video-based tasks. For examples, face recognition, classification and retrieval to name a few. In [1] a novel E-Commerce system has been proposed which is based on VGG-16 [2].

## 5. Application Specification and Implementation

### a) Workflow


Fashion DataSet
Initial 3-stage Image classifier module
Trained/formatted Array DataSet
Test Images
Actor
Modified 3-stage Image classifier module
Test/formatted Output Array
Output e-commerce recommendations and sites
Knowledge Recommendation Algorithm Module

b) System Specification

IDE: IntelliJ, SBT, Pycharm, Android Studio
Services: Image annotation - Clarifai API https://www.clarifai.com/, openimaj, Adobe spark, TensorFlow
https://spark.apache.org/, object detection - CS5542 lab tutorials.
Sources @ https://github.com/waynenguyen303/deep-learning-visual-eCommerce/tree/master/algorithm-sources

6. Project Management

a) Work completed/to be completed

Completed:
- Initial data set (hours = 3): Wayne Nguyen
- Algorithms research (hours = 5): all members
- Algorithm usage/code (hours = 5): Zhaobin Zhang, Wayne Nguyen
- IDE application implementation (hours = 10): all members
- Paper Incrementation (hours = 2): Wayne Nguyen
- Image/data set categorization (hours = 3): Hongcheng Jiang
- Application design/workflow (hours =3): Fei Wu, Zhaobin Zhang
- Tensorflow Algorithm: Wayne Nguyen
- Clarifai machine learning: Wayne Nguyen, Hongcheng Jiang
- Paper Increment 2: Wayne Nguyen
- Participation: all members 25% each

To be Completed:
- Refine application design/workflow (hours = 3): Zhaobin Zhang
- Get bigger data set (hours = 5): Fei Wu
- Implement Algorithms for trained data array (hours = 8): Hongcheng Jiang
- Converge algorithms (3-stages) for output array (hours = 8)
- Start knowledge recommendation algorithm (hours = 5)
- Paper increment 3 (hours =2):
- UI implementation (hours = 5): Wayne Nguyen
- TensorFlow CNN :
- Shallow Learning object classification:

[1] Devashish Shankar, Sujay Narumanchi, Ananya H A, et al, Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce, https://arxiv.org/pdf/1703.02344.pdf, 2017.
[2] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, CVPR, 2014.
[3] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, Deep Learning, Nature, 2015.