# Multi Model Approach for Item Prediction Using Item Based Filtering and CNN To Extract Image Features

Anuhya Suri (B00902313)
Aryan Khetani (B00802732)
Shiva Shankar Pandillapalli (B00880049)
Siddharth Rajan (B00868654)

## 1. Introduction

Most online shopping recommendation systems continue to rely heavily on user reviews and product pricing as their primary strategy for determining the most likely product that consumers want to buy. In some ways, this is inefficient because there is a lot more that goes through a consumer's mind before deciding on a product. One of the most important influencing factors is the product image. For our project, we combine image features extracted from clothing images, such as cloth pattern, colour, and category, with features used in a traditional recommendation system to generate better and more personalised recommendations.

Although predicting similar items has been a focus of recent research, one relatively unexplored area in this field is the incorporation of data associated with the product image. Shopping websites that use this methodology base their results primarily on data points such as product reviews and prices. To recommend similar products, we conducted research into what other factors, besides the review and the product price, could influence a user's decision. We discovered three such features that we believed could be important in determining that, namely cloth pattern, clothing colour, and cloth category.

We are motivated to work on this problem for two reasons. First, we want to make it easier for consumers to get better recommendations so they can shop more wisely. Second, our model can assist businesses in developing a better recommendation algorithm, which will eventually assist them in attracting more customers.

## 2. Related Work

Previous research has revealed a strong interest in improving and streamlining item recommendations.

Li Yu et al proposed a system for recommending items photos based on image content [1]. In this methodology, the photograph is first pre-processed, with the backdrop removed. Second, to represent the image, a weighted representation model is provided. The weights of each feature are computed based on the samples browsed by the users, after the separated features have been extracted and normalised. Third, based on the proposed representation, a feature indexing strategy is proposed. The algorithm's results on a real-world goods picture database demonstrated that it could recommend similar products photos with high accuracy and speed.

In a similar approach S. Muthamil Selvan et al suggested a Movie Recommendation Based on Posters and Still Frames [2]. In the paper they used Fully Connected Neural Networks to identify different movies with similar posters or stills in order to recommend them to users.

## 3. Dataset and Features

Data from the Deepfashion2 fashion dataset was used. It contains 491K images of 13 popular clothing categories from commercial shopping stores as well as consumers. We created a fictitious dataset for user-related data. The parameters in our dataset included the user's id, product reviews, and the number of products purchased by that user. The other dataset we used was the product dataset; we filtered out images from the deepfashion2 dataset that only had one garment and were clicked by the shopkeeper (instead of the user). Our product information dataset included two characteristics: product pattern and colour. We used a CNN model to identify the product pattern, and an OpenCV module to identify the colour. Finally, we used the following features for our recommendation system.

- As part of product details, we used cloth pattern, product colour, cloth category, occlusion, zoom and viewpoint.
- As part of the user details, we used user id, product reviews and number of products.

## 4. Methods

### 4.1 CNN

Convolutional Neural Network (CNN) [3] is a class of deep neural networks and is used for image processing. Convolution layers, pooling layers, and fully linked layers are among the layers that make up CNN. It learns spatial hierarchies of data automatically and adaptively using a backpropagation technique. The nonlinear activation function [8] ReLU (rectified linear unit) was chosen in the experiment because it is less susceptible to vanishing gradients, which limit deep model training. ReLU uses max(0.0,x) which just means to return a value of 0.0 instead of a negative value. The major drawback with using this activation function is the generation of saturated units.

The CNN model's accuracy was improved by using the activation function and altering weights and kernels. The CNN model that we use takes the input image and outputs the image's design feature. After that, the output is compared to the labels that were manually applied to the photos for training. The total loss is calculated by comparing the output with the picture label. The forward propagation loss is then employed in back propagation, which updates the model and improves accuracy. The model also uses dropout to prevent overfitting.

The training data had labels for the design pattern for the image. The labels were categorized amongst these categories: animal', 'cartoon', 'letter_numb', 'plain', 'printed' 'squares', 'stripes', and 'other'. Based on the accuracy of the CNN model, the weights are adjusted for each of the above categories. The CNN model had an accuracy of 70%.

### 4.2 Open CV

We used Open CV to identify the colours of the clothing items. Colors are made up of 3 primary colors; red, green, and blue. On computers, we define each colour value within a range of 0 to 255. In our approach, we take the cropped product image based on the bounding box data present in the dataset. We then use all the pixel values to calculate the mean RGB for the entire image. The mean value is then mapped to an existing list of colours with their RGB values and the colour having the closest value to our sample is returned (we have used about 55 colours in contrast to the small number of colours generally used).

### 4.3 Content Based Filtering

The content-based filtering strategy is used in our recommender system. The content-based recommender system generates new recommendations based on various product qualities. The following product attributes are considered in our dataset: product category (short sleeve dress, vest, etc.), dress colour, dress pattern (e.g., printed, plain, etc.), occlusion, zoom in, and viewpoint. The steps for creating content-based filtering are as follows:

1. A user dataset is built, which includes mappings between people, products, and the ratings given to each of them. There are 100 users and a total of 10,000 products. A maximum of 80 products are randomly assigned to each user.
2. The recommendation system's target variable is the product reviews column in the user dataset [1]. For simple comparison, it is normalised to values between 0 and 1.

3. The product properties dataset is constructed, which contains the product pattern (output from CNN), colour (output from CV2), category, occlusion, viewpoint, and zoom in (these variables are directly present in the original dataset)
4. Because the attributes considered in this approach are all categorical, the vector representation for the products is obtained using term frequency inverse document frequency (TF-IDF) [3]. The TF-IDF approach is utilised because it represents the significance of each value by capturing the frequency of each value in the categorical variables.
5. The cosine similarity metric is used to calculate the similarity between the products for each attribute. To generate the final product similarity matrix, the cosine similarity matrices of all attributes are merged by element wise product [4].
6. The product reviews column is used to create a user-product matrix. Matrix multiplication is used to combine the user-product matrix and the product-product similarity matrix to get weighted rating values for each user-product combination.
7. The results are standardised and compared to the original standardised ratings.
8. This strategy aids in predicting the ratings a customer will give a product they have never purchased before. The user can be recommended the top five or ten products with the highest anticipated ratings.

## 5. Workflow

Initially, a variety of datasets were explored for the experiment. DeepFashion2 was found to be the best dataset for our experiment since it included several fields such as Bounding Box, which were crucial for accurate feature extraction. The DeepFashion2 dataset, on the other hand, contains photographs that were not captured properly and that would reduce the training accuracy. For feature extraction to be possible, the dataset required to be filtered using viewable photos.

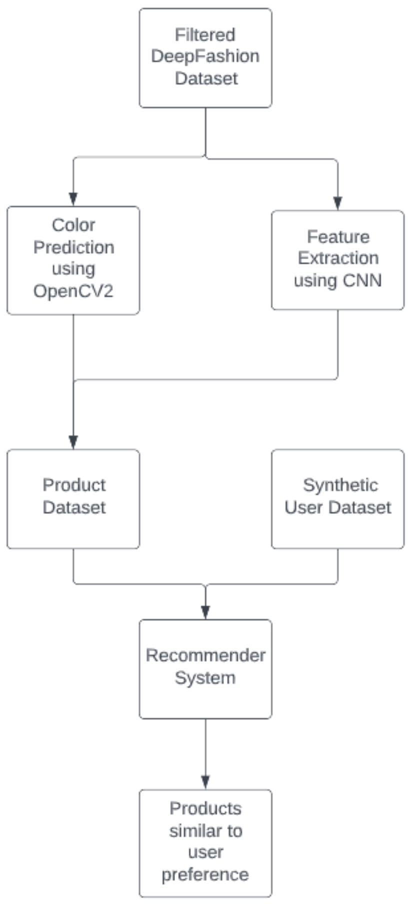

Filtered DeepFashion Dataset
Color Prediction using OpenCV2
Feature Extraction using CNN
Product Dataset
Synthetic User Dataset
Recommender System
Products similar to user preference

**Figure 5.1: Workflow of the models**

The dataset contained a lot of features, but it lacked critical ones like colour and design type, which are used to recommend items. The primary colour of the products was retrieved using OpenCV2. The pattern features were then extracted using a convolutional neural network. To test the CNN model, the pattern feature categories were manually annotated. Thereafter, the model was trained and added to the product dataset. A user-specific dataset was synthetically generated to be used in the recommender system.

Both the product and user databases are incorporated into the recommender system. The recommender system examines the items purchased by a certain user and based on those purchases, proposes comparable products from the datasets that were passed.

## 6. Experiments

### 6.1 Convolutional Neural Network

The purpose was to extract pattern information from the deepfashion2 dataset because we planned to use it in content-based filtering to provide recommendations for end-users. To achieve this on the deepfashion2 dataset, which lacks the needed labels, we created a neural network model that was trained on labelled images in another clothing pattern recognition dataset [3], with the goal of predicting the labels on the deepfashion2 dataset with high accuracy. Both of our trials, predicting the labels using the model we trained on the dress pattern recognition dataset and using the pre-trained model that has an accuracy of around 98 percent on the dress pattern recognition dataset, failed to provide us with satisfactory results. The poor outcomes were due to quality of the images.

As a result, to train the model to label the photos, we chose to manually annotate 1600 randomly selected images from the dataset shared among us. We re-ran our trails after annotation to determine if the annotated photos' accuracy had increased. However, we ran across an unanticipated situation where our model performed admirably on individual data but not on collective data. We analyzed the situation and came to the conclusion that the problem was caused by our perspectives. Then one of us went over all of the photographs and made sure they were correct.

Even after examining and reannotating all of the images, the accuracy did not improve as much as anticipated. We assumed that modifying the network structure might help us achieve better results, so we increased the number of conv2D, pooling layers from two to four, and the number of affine layers from two to seven. We tried every possible combination and came to the conclusion that they all performed similarly. Infact, network with more affine layers was giving considerably less accuracy. Among all possible combinations, the network with two conv2D layers and three affine layers performed well. As a result, we decided to go forward with it. During the construction of this network, we employed Stochastic Gradient Descent as an optimizer and Relu as an activation function as sigmoid did not give us satisfying results in earlier experiments.

At one point, we felt that retraining the same model on the same data would solve the accuracy problem. However, we quickly saw that our assumptions were incorrect, as the model became overfitted for the data and performed poorly on the testing data. That's when we started thinking about hyperparameters. We began by experimenting with batch sizes, increasing and decreasing them. Following various tests, we decided on a batch size of four because it produced good results on the dataset.

We attempted to run as many epochs as possible based on the dataset. We started with five epochs and trained the network for roughly two hundred epochs. We couldn't seem to find a good spot, so we experimented with different epochs for each of the hyperparameters we chose. We experimented with the learning rate and discovered that increasing it to 0.1 causes the network to learn more quickly, but we soon realised that the network is missing important features from the photos. So, based on the size of the dataset and the epochs, we saw that 0.01 and 0.005 were conducting descent after numerous trials with a learning rate ranging from 0.0001 to 0.1. As a result, we relied on these two. We also noticed that as the learning rate increased, the cost of a few epochs started increasing suddenly.

To further understand how the model works, we attempted training and testing splits of 70:30, 80:20, and 90:10 for individual and mixed dataset sizes (400 to 1600 photos). According to our findings, our model functioned admirably for all ratios. When the training sample size was increased, the model performed well. As a result, with the exception of a few instances, we opted to employ a 90:10 ratio most of the time.

We also experimented with the Tanh and Sigmoid activation functions. However, we read that Relu is the only activation function that performs better with convolutional neural networks, according to this article [7]. When we used the sigmoid activation function, we found that the cost was not lowered but the accuracy was. When compared to the sigmoid activation function, utilising Tanh reduced the cost and, interestingly, boosted the accuracy. Tanh appears to be working as a result of the derivatives in the backward pass.

We opted to normalise the data after many trials because we know that normalising the data will stabilise the network and gradients, which perform better in convergence than normal. Even still, normalising the data did not help us; we believe that normalising the data may impair the network's ability to recognise patterns. Batch norm, in our opinion, is working because it uses the mean and standard deviation of the entire batch rather than a single sample, which allows the network to perform effectively when data is sent to the next layer because it is stabilised.

The addition of the batch norm layer has prompted us to consider adding a dropout layer as well, to turn off the neurons and assess how well the network performs. We experimented with odds of 0.25 and 0.50. Although neither of them provided stunning results, the network with a dropout layer with a probability of 0.5 produced pleasing results because, as mentioned in the co-adaptation problem [8], noise can occasionally be decreased by turning a few neurons. As a result, we decided to go ahead with it.

We also attempted to accelerate network convergence by injecting momentum, anticipating that the network would learn more quickly. We were able to speed up the network convergence, but this did not increase our accuracy. We decided to use Adam instead of SGD since it works with the mean of the pixels in terms of gradients and learning rates per parameter, as indicated in article [6]. We noticed the shift since it converged rapidly and provided acceptable accuracy even with as few as five epochs. However, as compared to SGD, the accuracy was not greater because the patterns were dependent on pixel values, which were not learned adequately due to Adam optimizer's methodology.

Finally, after several tries with all of the above-mentioned hyper parameters and various layers, we discovered a combination that worked well in obtaining accuracy of around 70% and occasionally higher. The following is the model.

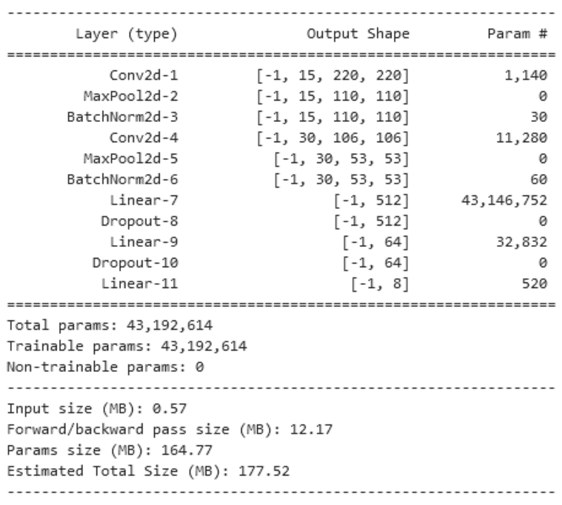

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 15, 220, 220]           1,140
         MaxPool2d-2         [-1, 15, 110, 110]               0
       BatchNorm2d-3         [-1, 15, 110, 110]              30
            Conv2d-4         [-1, 30, 106, 106]          11,280
         MaxPool2d-5           [-1, 30, 53, 53]               0
       BatchNorm2d-6           [-1, 30, 53, 53]              60
            Linear-7                  [-1, 512]      43,146,752
           Dropout-8                  [-1, 512]               0
            Linear-9                   [-1, 64]          32,832
          Dropout-10                   [-1, 64]               0
           Linear-11                    [-1, 8]             520
================================================================
Total params: 43,192,614
Trainable params: 43,192,614
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 12.17
Params size (MB): 164.77
Estimated Total Size (MB): 177.52
----------------------------------------------------------------
```

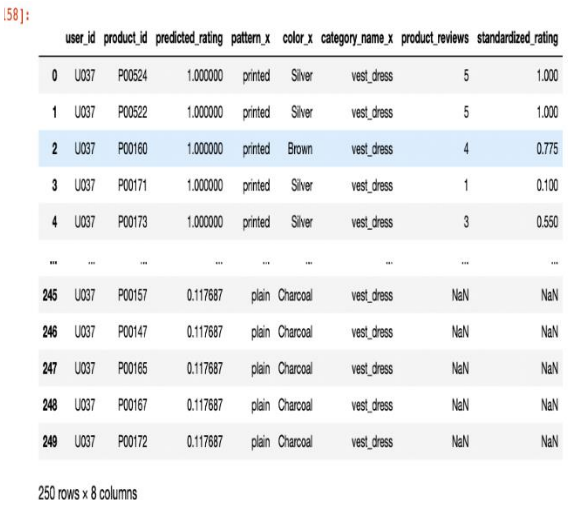
**Figure 6.1: CNN Model**

## 6.2 Content-based Filtering

For content-based filtering, category, pattern, and colour are initially employed as attributes. An example result from this experiment is shown below.

As shown in **Figure 6.2.1**, U037 has given ratings to three products with the same pattern = 'printed,' colour = 'Silver,' and category name ='vest dress,' namely P00524, P00522, and P00171. User U037's initial ratings for the first two products are 5,5 (standardized rating = 1,1,0.1). For the third product, the rating given is 1. However, because the three products have the same features, the predicted rating for these products is 5,5,5.

(Note that the predicted rating column is normalised, thus 1.0000 denotes a 5-star rating.) As a result, the three characteristics alone are insufficient to explain the scores.

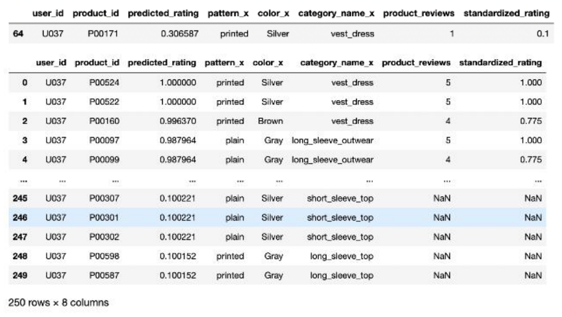

[58]:

| | user_id | product_id | predicted_rating | pattern_x | color_x | category_name_x | product_reviews | standardized_rating |
|---|---|---|---|---|---|---|---|---|
| 0 | U037 | P00524 | 1.000000 | printed | Silver | vest_dress | 5 | 1.000 |
| 1 | U037 | P00522 | 1.000000 | printed | Silver | vest_dress | 5 | 1.000 |
| 2 | U037 | P00160 | 1.000000 | printed | Brown | vest_dress | 4 | 0.775 |
| 3 | U037 | P00171 | 1.000000 | printed | Silver | vest_dress | 1 | 0.100 |
| 4 | U037 | P00173 | 1.000000 | printed | Silver | vest_dress | 3 | 0.550 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 245 | U037 | P00157 | 0.117687 | plain | Charcoal | vest_dress | NaN | NaN |
| 246 | U037 | P00147 | 0.117687 | plain | Charcoal | vest_dress | NaN | NaN |
| 247 | U037 | P00165 | 0.117687 | plain | Charcoal | vest_dress | NaN | NaN |
| 248 | U037 | P00167 | 0.117687 | plain | Charcoal | vest_dress | NaN | NaN |
| 249 | U037 | P00172 | 0.117687 | plain | Charcoal | vest_dress | NaN | NaN |

250 rows x 8 columns

**Figure 6.2.1: Experiment-1 Results**

To account for this variation, the following experiment incorporates occlusion, viewpoint, and zoom in in addition to the current variables. The addition of these variables improved the accuracy of the ratings prediction. The results of the second experiment with the same user and product are shown below.



| | user_id | product_id | predicted_rating | pattern_x | color_x | category_name_x | product_reviews | standardized_rating |
|---|---|---|---|---|---|---|---|---|
| 64 | U037 | P00171 | 0.306587 | printed | Silver | vest_dress | 1 | 0.1 |

| | user_id | product_id | predicted_rating | pattern_x | color_x | category_name_x | product_reviews | standardized_rating |
|---|---|---|---|---|---|---|---|---|
| 0 | U037 | P00524 | 1.000000 | printed | Silver | vest_dress | 5 | 1.000 |
| 1 | U037 | P00522 | 1.000000 | printed | Silver | vest_dress | 5 | 1.000 |
| 2 | U037 | P00160 | 0.996370 | printed | Brown | vest_dress | 4 | 0.775 |
| 3 | U037 | P00097 | 0.987964 | plain | Gray | long_sleeve_outwear | 5 | 1.000 |
| 4 | U037 | P00099 | 0.987964 | plain | Gray | long_sleeve_outwear | 4 | 0.775 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 245 | U037 | P00307 | 0.100221 | plain | Silver | short_sleeve_top | NaN | NaN |
| 246 | U037 | P00301 | 0.100221 | plain | Silver | short_sleeve_top | NaN | NaN |
| 247 | U037 | P00302 | 0.100221 | plain | Silver | short_sleeve_top | NaN | NaN |
| 248 | U037 | P00598 | 0.100152 | printed | Gray | long_sleeve_top | NaN | NaN |
| 249 | U037 | P00587 | 0.100152 | printed | Gray | long_sleeve_top | NaN | NaN |

250 rows × 8 columns

**Figure 6.2.2: Experiment-2 Results**

As per Figure 6.2.2, we can see that the predicted rating for the third product P00171 has altered from 1.00 to 0.306 because of this experiment. As a result, the addition of the new attributes enhanced the rating predictions.

Apart from the six attributes used for content-based filtering, other factors such as the product's brand, price, and so on can influence the ratings given by each user.

## 7. Results

All categories made significant contributions to the prediction of similar items. We were able to achieve an overall accuracy of 70% by feeding information about the product image into the recommendation system. Although we don't have a baseline against which to compare our accuracies, the accuracy of the recommendation system improved as we added more features. Among all of the features, one that we extracted using a CNN model was the product pattern. The model performed admirably, with an overall accuracy of 74%. As a result, for item recommendation, production companies and consumers should consider various product image features. Adding new features, such as cloth pattern, fabric type, and occlusion, could significantly improve the item recommendation. Future research could focus on the creation of a sophisticated nested model to extract more complex features from the image which would enhance the recommendation system even further.

## 8. References

[1] Selvan, S., Pudhota, M., Gunnam, S. and GP, K., 2022. *Movie Recommendation Based on Posters and Still Frames using Machine Learning*. [online] Ijeat.org. Available at: <https://www.ijeat.org/wp-content/uploads/papers/v9i4/D7255049420.pdf> [Accessed 17 April 2022].

[2] L. Yu, Y. Luo, F. Han and S. Huang, "A content-based goods image recommendation system", link.springer.com, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s11042-017-4542-z. [Accessed: 17- Apr-2022]

[3] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9,

no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/s13244-018-0639-9.

[4] Jhawar, A., 2022. GitHub - aakashjhawar/dress-pattern-recognition-using-CNN: An image recognition model which is capable of identifying the pattern on a dress image. [online] GitHub. Available at: <https://github.com/aakashjhawar/dress-pattern-recognition-using-CNN> [Accessed 17 April 2022].

[5] B. Allison, "Is there a rule-of-thumb for how to divide a dataset into training and validation sets?", Stack Overflow, 2022. [Online]. Available: https://stackoverflow.com/questions/13610074/is-there-a-rule-of-thumb-for-how-to-divide-a-dataset-into-training-and-validatio. [Accessed: 17- Apr- 2022]

[6] Pytorch.org. 2022. Training a Classifier — PyTorch Tutorials 1.11.0+cu102 documentation. [online] Available at: <https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html> [Accessed 17 April 2022]

[7] Kingma, D. and Ba, J., 2022. Adam: A Method for Stochastic Optimization. [online] arXiv.org. Available at: <https://arxiv.org/abs/1412.6980> [Accessed 17 April 2022].

[8] Brownlee, J., 2022. How to Choose an Activation Function for Deep Learning. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/> [Accessed 17 April 2022].

[9] Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2022. Improving neural networks by preventing co-adaptation of feature detectors. [online] arXiv.org. Available at: <https://arxiv.org/abs/1207.0580> [Accessed 17 April 2022].

[10] "GitHub - switchablenorms/DeepFashion2: DeepFashion2 Dataset https://arxiv.org/pdf/1901.07973.pdf", GitHub, 2022. [Online]. Available: https://github.com/switchablenorms/DeepFashion2. [Accessed: 17- Apr- 2022]

[11] Bushaev, V., 2022. Stochastic Gradient Descent with momentum. [online] Medium. Available at: <https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d> [Accessed 17 April 2022].

[12] "Conv2d — PyTorch 1.11.0 documentation", Pytorch.org, 2022. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html. [Accessed: 17- Apr- 2022]

[13] "MaxPool2d — PyTorch 1.11.0 documentation", Pytorch.org, 2022. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html. [Accessed: 17- Apr- 2022]

[14] "NumPy quickstart — NumPy v1.22 Manual", Numpy.org, 2022. [Online]. Available: https://numpy.org/doc/stable/user/quickstart.html. [Accessed: 17- Apr- 2022]

[15] "pandas - Python Data Analysis Library", Pandas.pydata.org, 2022. [Online]. Available: https://pandas.pydata.org/. [Accessed: 17- Apr- 2022]

[16] P. Fork, "Pillow/Image.rst at 5d070222d21138d2ead002fd33fdf5adcb708941 · python-pillow/Pillow", GitHub, 2022. [Online]. Available: https://github.com/python-pillow/Pillow/blob/5d070222d21138d2ead002fd33fdf5adcb708941/docs/reference/Image.rst. [Accessed: 17- Apr- 2022]

[17] Matplotlib Developers, 2022. Matplotlib — Visualization with Python. [online] Matplotlib.org. Available at: <https://matplotlib.org/> [Accessed 17 April 2022].

[18] D. Chen, “Recommender System — Matrix Factorization - Towards Data Science,” *Medium*, Jul. 08, 2020. https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b.

[19] Àlex Escolà Nixon, “Building a memory based collaborative filtering recommender,” *Medium*, Oct. 29, 2020. https://towardsdatascience.com/ho

w-does-collaborative-filtering-work-da56ea94e331.

[20] Àlex Escolà Nixon, “Building a movie content based recommender using tf-idf,” *Medium*, Aug. 28, 2020. https://towardsdatascience.com/content-based-recommender-systems-28a1dbd858f5.

[21] Tan Pengshi Alvin, “A Content-Based Recommender for E-Commerce Web Store,” *Medium*, Nov. 16, 2020. https://towardsdatascience.com/a-content-based-recommender-for-e-commerce-web-store-7554b5b73eac.