Fatching: Deep Learning Based Fashion Recommendation Application

Hangyu Kim, Bomin Namkoong, Juwon Suh,
Gyeonghyeon Cho, Jaehyuk Choi

December 2021

Abstract

As data becomes bigger and bigger everyday, people are searching for various tools to process information. Deep learning technology enables processing such data systematically. In this paper, we would like to propose an application using deep learning to enhance the convenience of users.

In particular, the application recommends other fashion items that match well with the fashion item the user already has. User can upload picture, search gallery or even just select detail options of the clothes without having to actually upload the certain item. Through app "Fatching", user can decide on their outfit of the day within one second.

Keywords— deep learning, clothes classification, fashion recommendation

1 Introduction

Fashion industry has shown continuous growth in its market size. The overall rate of fashion market has grown by about 10 trillion won over the past decade. This increase in the market led to expansion in all sorts of fashion area. Distribution of fashion is now available not just in offline markets but is rather active in online, mobile shopping. Various technology related services are expanding and the use of machine-learning for customized service is very common nowadays. Furthermore, fashion products are getting much more diverse to satisfy a wide range of customers.

There are some downside to the current fashion market phenomena. Due to the excessive fashion information provided by numerous fashion media and the rapidly changing nature of fashion trends as time goes by, many people have difficulty deciding which combination to wear, or simply which clothing goes well with others. To help solve this problem, some online fashion stores and communities provide customized coordinates or products. However, this information is one-sided because they operate on pre-set photos of professional models and simply recommend products with highest selling without a comparison group. In our project, we aim to provide better service

for users to find a fine combination of clothing items.

In this project, we classify fashion images through the labeling data of fashion image dataset. For example, a standard fit white shirt will be labeled as white, top, midi-length, standard-fit clothing. Once the classification is completed, we will implement a deep learning algorithm that uses K-fashion images to determine how many clothes are combined with a particular item. Users can simply run application "Fatching" and search best combination of their fashion item by taking a photo, going through gallery or even just selecting item options of their clothing. Customers will be recommended fashion combination simply by clicking a few times in the application.

2 Related Work

2.1 Related Works on Fashion Recommendation Service

Content-based filtering recommends appropriate styles by applying customer's personal information to product profiles. China's Alibaba Group identifies users with two-dimensional codes and facial recognition based on the big data of 500,000 Taobao members, and recommends products based on their existing purchase history and tastes.[1] Zozo town, which creates a body profile by ap- plying artificial intelligence measurement technology, uses 3D body scan tech- nology to measure body size so that when customers purchase clothes, they skip the process of visiting a store, fitting, and worrying about size. Instead, it make possible to select various styles of products suitable for the body type of people.[2]
Recently, fashion companies are providing more advanced image tag ser- vices through partnerships with IT companies. Amazon's Echo look recognizes the user's wearing photo through the artificial intelligence system Alexa, and quantifies the results of machine learning on fit, color, and styling and results on fashion styling based on current trends.[3] Furthermore, Coded Couture, introduced by Google, utilizes GAN(Generative Adversarial Network), which is attracting attention as a next-generation deep learning algorithm, to collect smartphone data such as climate, lifestyle, and frequently visited places of the user's residence for one week. After that, Coded Couture has put into practice a service that produces personalized clothing based on the collected data.[4]

2.2 Difference from Prior Works

We wanted to implement the recommendation part as deep learning. However the difficulty was high, so we came up with another solution. This is a method of recommending the highest frequency combination to users after producing 432-by-432 matrix based on the fashion vector, which is the output of the classification model. This method is simpler than the method using deep learning, but ensures high recommended accuracy.

3 Proposed System/Solution/Service

3.1 Overall Structure

The proposed system of project Fatching is consisted as the figure below. Frontend uitilized android to build UI/UX and server processed data through python, Flask and Tensorflow. In backend, classification and recommendation model were built through Tensorflow and numpy.

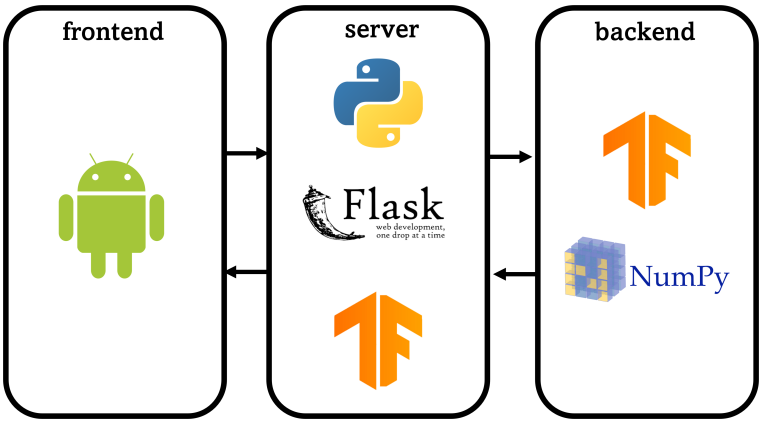

Figure 1: frontend-server-backend

3.2 Core model

There are 2 core models in Fatching. One is fashion classification model and the other is fashion recommendation model. Since details of these models will be discussed in the implementation part, the overall flow of data processing will be mentioned in this section. When the user inputs a photofile to the application, classification model determines the type of clothes based on four criteria: color, category, length, and fit. Once the image is classified, the program needs to provide a recommendation for the given item. Fatching does this by analyzing how many times a item is matched to another in our certain dataset. This allows the service to recommend a combination of clothes that are frequently matched together.

3.3 General challenges

There were various difficulties while implementing the project. We handled each problem by taking diverse approaches.

3.3.1 Backend

Accuracy problem was one of the most important issues to deal with in Backend. After trying out various methods, we were able to solve problem through random transformation. That is applying randomized transformation such as image rotation

through data augmentation to increase the diversity of training sets. Since more aspects of dataset is identified, this led to data enhancement.

3.3.2 Frontend

In frontend, implementing item options bar was a challenge. Because android supports various tools to construct UX/UI, it is important to choose the right tool for the desired function. Option bars in Fatching was first implemented in the most basic tool, which is popup menu. However, popup menu wasn't capable of displaying the selected menu on a fixed textview. So instead, item options were implemented through spinners. This made application Fatching look the way it is now.

4 Design

Our app Fatching paid attention to designing to provide users with the easiest and most intuitive experiences through their daily lives. Our ultimate goal was to ensure that users did not spend much time and worry about choosing their daily look. Therefore, our design focused on minimizing the user's touch required to implement the function within the UI as simple as possible.

4.1 Frontend Design

The app UI consists of a total of three screens: the main page, the advanced page, and the Result page.The main page and advanced page are pages where users upload items and send them to the server, and the result page is a page where items that match the uploaded items are recommended.

4.1.1 Main page

Figure 2: main page

When you run the Fatching app, it is the first page to run. Click on the purple box in the center of the screen to launch the gallery app and upload the photo file. Click

on the gray letter "Advanced" in the center of the purple box to move to the advanced page where more detailed search is possible. When the camera icon is clicked, the camera application is executed, and the item may be immediately taken and uploaded. In the 'Style' spinner, you can select three styles: Mannish, Modern, and Military. When you click the Search button, go to the Result page and show the recommended results.

4.1.2 Advanced page

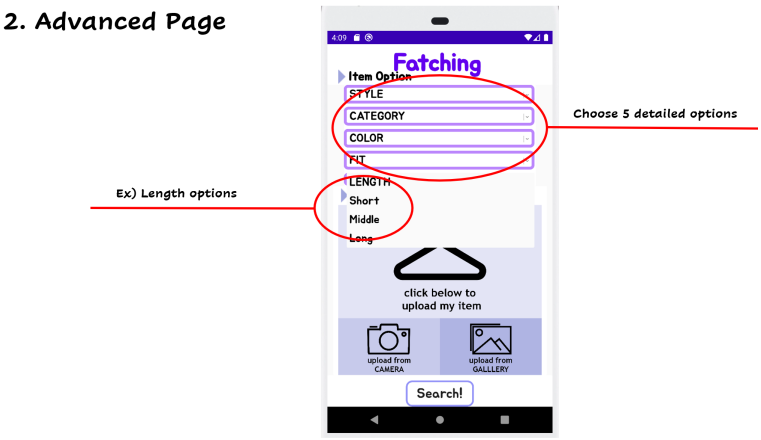

Figure 3: advanced page

It is an advanced page that can be searched with a more detailed option. On this page, Style (sub-option: Mannish, Military, Modern), Category (sub-option: Top, Bottom, Outer, Dress), Color (sub-option: Black, Blue, Brown, Green, Gray, Orange, Pink, Red, Violet, White), Fit (Tight, Normal), Like the main page, you can retrieve and upload photos from the camera (left) and gallery (right) through the two buttons at the bottom. You can check the recommended items on the result page by pressing the Search button.

4.1.3 Result page

The Result page is a page that can be checked by outputting recommended items that match the picture uploaded by the user. First, the user clicks the 'Click' button to check the recommended item. Then, recommended items are output in the six purple box layouts in the center of the screen. Check the output items and click the Another search button at the bottom of the screen to return to the main page, the first page, and explore new items.

4.2 Backend Design

The backend part of Fatching is divided into two categories: a fashion classification model and a fashion recommendation model. First, the fashion classification model is

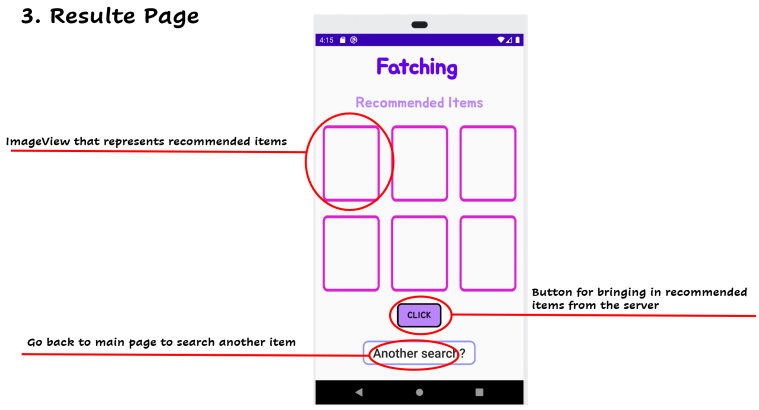

Figure 4: result page

a model that receives user's input image data and accurately classifies what type of clothes it is based on four criteria. To create this model, we used Tensorflow, a machine learning library released by Google as an open source. We decided to use Tensorflow for model implementation, because Tensorflow provides various functions to make it easy for general users to use deep learning and machine learning, and it could make us to write computation processing code using python that our team can easily use best. Also, we imported and used Tensorflow's keras, which uses a deep learning library as a backend to easily construct neural network models and combinatorial models. Using these functions, we developed a total of 10 fashion classification models that classify fashion images according to four criteria of color, category, length, and fit and pass them on to fashion recommendation model.

Next, the fashion recommendation model is a model that tells which item is best matched with user's input image data based on an extensive dataset using classified clothing information. This model was implemented using Numpy, a python package for mathematical and scientific operations. Numpy processes repetitive operations such as for statements in an array unit, thus increasing the actual operation speed and enabling efficient coding. We created an algorithm that creates a fashion recommendation array using this Numpy and performs indexing, processing, and operation based on it. In this array, the user's input image data classified by color, category, length, and fit according to the fashion classification model finds the most items in the corresponding column and returns its type and image. This array can be saved and loaded, so it can be updated by continuously adding data.

5 Implementation

5.1 Frontend Implementation

Frontend was developed using the kotlin language within Android Studio IDE. There are three pages in the Fatching app, and each page has to perform various

functions such as camera, gallery, spinner, and photo upload, making it difficult to organize complex layouts and place view widgets only through layout configuration through activity. Therefore, we chose a method of separating and managing layouts and reducing the complexity of layouts by dividing the elements of each function of the screen into fragment units. For each screen of the Fatching app, the main page was developed through BlankFragment, the advanced page through NextFragment, and the result page through SearchFragment. Android Navigator library was used to switch screens of each fragment through Search button. Through this, the screen switching process was created into an xml file such as layout resouce, thereby obtaining the advantages of visualization, maintenance and transaction processing of the fragment could be facilitated. Between fragments, data such as uploaded photo paths were transferred through bundle.

Particularly difficult point occurred when uploading photos from galleries and cameras, respectively. From Android version 7.0, content Uri was inevitably used because it was not recommended to use file Uri, which is directly accessible to internal storage, due to security issues in the process of converting Uri from gallery to Filepath. However, since we needed the absolute path of the file, we solved it by parsing the absolute path through query in Media database. In addition, in the process of handing over bitmap information received from photos taken with a camera, the getBitmap method was deprecated and could not be used at versions with SDK version higher than 28. We solved this problem using the decodeBitmap method of the ImageDecoder class.

5.2 Backend Implementation

5.2.1 Fashion Classification Model

The fashion classification model is a model that receives user's input image data and accurately classifies what type of clothes it is based on four criteria. As the dataset for the fashion classification model, K-Fashion image dataset, one of AI-Hub's vision domain open data, was used. The K-Fashion image dataset consists of 1.2 million images and includes labeling information so that fashion areas, attributes, and style information can be easily recognized. All images in the dataset have their own labeling data, and we decided to make a model that classifies the user's input image according to four criteria using color, category, length, and fit information among the labeling data.

The dataset for learning the color classification model is divided into 12 colors, and each color consists of 25 corresponding color images. The dataset for learning the category classification model is divided into four categories: bottom, dress, outer, and top, and each consists of about 3,000 images. The dataset for learning the length classification model is divided into long, middle, and short in the 4 categories, and consists of about 1000 images each. The dataset for learning the fit classification model is divided into loose, normal, and tight in the 4 categories, and consists of about 1000 images each.

The biggest challenge while constructing the dataset is that when the completed model performs category classification, it often outputs incorrect results by distinguishing the input data by color rather than the shape of clothes. Therefore, the original image was initially used as it was for learning, but in order to remove the result of color, the process of converting the image to grayscale was additionally per-

formed and this data was changed into a dataset for model learning. As a result, it was possible to increase the training accuracy and validation accuracy of the category classification model by about 8 percentages and 5 percentages, respectively, as shown in the figure below.

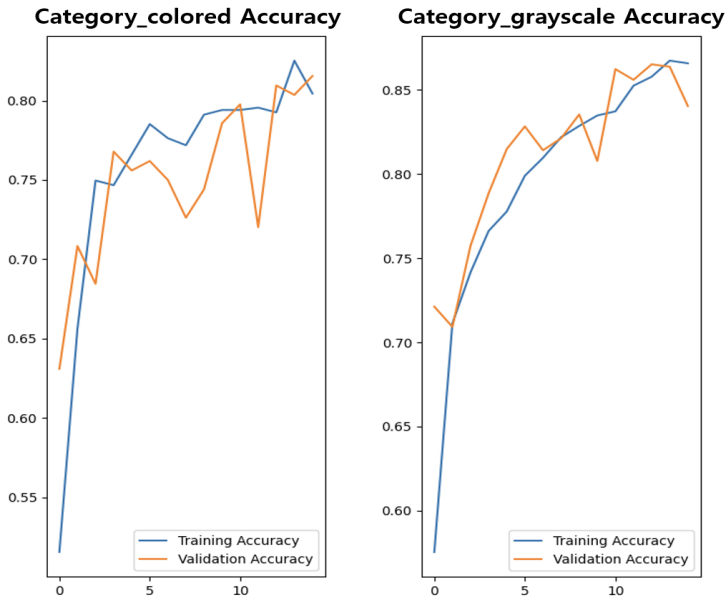

Figure 5: Colored Accuracy and Grayscale Accuracy

As mentioned in the design part, Tensorflow was used to learn the fashion classification model. The loader parameter batch size for model learning was set to 32, and 80 percentages of the dataset image was used for training and the remaining 20 percentages was used for verification to use verification segmentation. The epoch for model learning was set to 15.

In addition, we used Cache method to allow dataset to be cached in memory or local storage to further increase the performance of the model. When caching a dataset using the Cache method, the task of reading the dataset image is executed only during the first epoch and the data cached by Cache method conversion is reused from the next epoch. Therefore, during model training, the dataset does not become a bottleneck, and a situation in which the dataset is too large or memory is insufficient can be solved by generating a high-performance cache. Also, random transformation such as image rotation was applied through data augmentation to increase the diversity of training sets. Generalization, that is, accuracy, increases as the model identifies more aspects of the dataset through data augmentation.

Using the labeling data, we created one model each to classify color and category, and four models each to classify length and fit according to the previously classified category. Therefore, a total of 10 learning models were created through all of the

above processes, and the accuracy of each model reached a minimum of 70 percentages to a maximum of 90 percentages. As a result, when the user uploads a fashion image to the application, the image data is classified into four criteria through the fashion classification model stored in the server, and this data is moved to the fashion recommendation model to be mentioned in the next section.

5.2.2 Fashion Recommendation Model

The fashion recommendation model is a model that tells which item is best matched with user's input image data based on an extensive dataset using classified clothing information. The fashion recommendation model uses a Numpy array to store data and returns a recommendation result based on the classified results. Same as the classification model, we used the dataset for fashion recommendation model consisting of 12 colors, 4 categories, 3 lengths, and 4 fit. As a result, there are 432 possible combinations, and that's why the recommendation array is made in the form of 432 by 432.

First, we made 432 by 432 zero matrix to connect all cases of fashion vectors. And then, add 1 to the cell that matched each other. For example, if black pants and yellow shirts match, add 1 to the corresponding compartment. After analyzing all the dataset and updating the matrix, the algorithm will use the fashion vector that was the output of the classification model to select an item that goes well with the fashion vector, that is, an item with the highest value in the matrix, and send it to the application.

In order to generate array data for fashion recommendation model, one set was formed by dividing AI-Hub and other various fashion images into four categories: top, bottom, outer, and dress. In order to create the set, only images showing all the body were used, and all items in one set were classified using fashion classification models, and the results were stored in the array. The completed array is called to the server and the classification result of the input data by the user is substituted. The array then finds and returns up to three data lines that match the most with the classification results, and shows these three results again to the user.

One challenge with the completed fashion recommendation model was that many of the dataset constituting the array were black clothes. Therefore, in the process of recommending matching items up to the third priority, there was a problem that all colors up to the third priority were fixed in black due to overwhelming black-based data. This was not an inappropriate or incorrect result due to the characteristics of the actual current fashion trend with the dataset, but the algorithm was changed to remove the recommended color from the previous ranking for the diversity of recommendation results.

6 Limitations and Discussions

First, in the process of training the fashion classification model, we encountered two limitations. First, in the process of training the dataset, we planned to improve the accuracy of the model as high as possible by performing data augmentation on the basic data and then transfer learning. However, when transfer learning is carried

out, the learning time is increased more than 20 times from 1 hour to 20 hours based on 10,000 images, so the transfer learning process has to be removed due to time constraints. For the same reason, we had 1.2 million fashion images provided by the AI-Hub, but there were many overlapping images, including examples of shooting a fashion style from various angles. Therefore, 12,000 images among all images were used due to the limitation of learning time realistically. If we solve the above two problems, we can further improve the accuracy of the fashion classification model, but since we have already obtained 70 90 percentages of accuracy, we judged that it is inefficient to invest more time in training the fashion classification model to further improve the accuracy.

Next, there was a limitation of the dataset in the process of creating a fashion recommendation model. First, due to the overwhelming number of black clothing mentioned in the previous implementation part, we proceeded to further modify the algorithm. In addition, since we directly composed the dataset, we were unable to significantly increase the diversity of the 432 output results because sufficient data could not be supplied even though we supplied the maximum amount of data as possible. This will have to be solved by continuously updating the array to add data to secure a large amount of array data.

7 Evaluation

7.1 Overall architecture of the Algorithm

7.1.1 Input and Output for the Algorithm

The dataset for the fashion classification model used the K-Fashion image, one of AI Hub's vision area opening data. It included labeling information to easily recognize fashion areas, attributes, and style information, so it was judged to be suitable as a clothing classification AI learning dataset.

The dataset for color classification model learning is divided into 12 colors, and each color consists of 25 corresponding color images and for category classification model learning, it is divided into four categories: bottom, dress, outer, and top, each consisting of about 3,000 images. The dataset for learning the Length and Fit classification models are divided into three cases in four categories, respectively, and consists of about 1,000 images each.

Same as the classification model, we use the dataset for recommendation model consisting of 12 colors, 4 categories, 3 lengths, and 4 fit. As a result, there are 432 possible combinations, and that's why the array is made in the form of 432 by 432.

7.1.2 Hyperparameter

The loader parameter batch size for model learning was set to 32, and 80% of the dataset image was used for training and the remaining 20% was used for verification to use verification segmentation. The epoch for model learning was set to 15.

7.2 Empirical Results

We ran the completed application and actually conducted the test.

First, when selecting and uploading a photo in the gallery in the first page and then going to the results page, it was confirmed that the appropriate recommendation results came from the user's point of view.

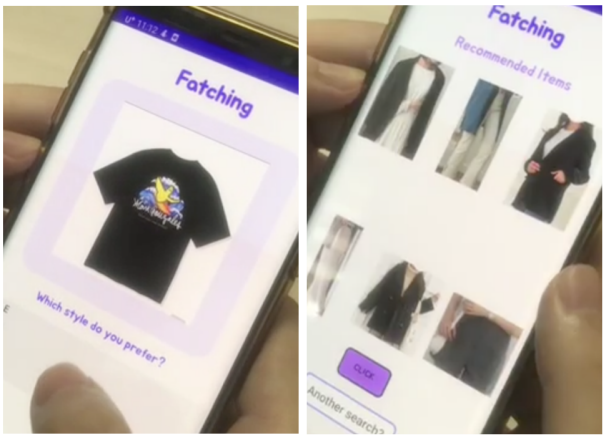

Figure 6: result page

Next, the advanced search page was tested. This time, after using a photo taken directly using a camera application, the user selected an option for clothes and proceeded, and it was confirmed that an appropriate recommendation result came out.

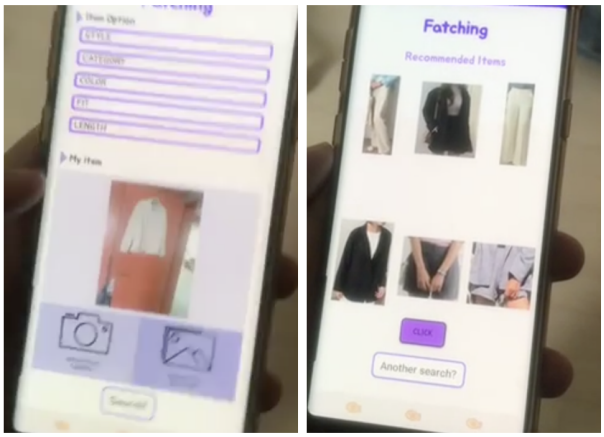

Figure 7: result page

7.3 Comprehensive Assessment of the Project

We succeeded in implementing an algorithm that recommends clothes suitable for the item held by the user, which was the initial implementation goal. Although it takes more than 5 seconds to press the search button and get the result, the recommended algorithm is in a state of working well, so it is judged that the initial goal has been achieved to some extent.

8 Conclusion

Fatching is a compound of fashion-matching, which means the fashion recommendation service. We supposed the idea to find the best outfit with the item we have. Actually, most fashion applications on the market don't have recommended services, and it is harder to find the clothes that go well with the item in our closet. Therefore, we designed the application to get some advice for our own clothes.

In the back-end, based on a wide range of datasets, it was developed into two parts: a fashion image classification model that accurately classifies what type of clothing it is by receiving input data, and a fashion recommendation model that find the best matching result. The former generates a fashion vector based on deep learning, and the latter produced a 432-by-432 matrix based on this to output the result value.

In the front-end, based on the flask, the application was created using Android Studio. There are three pages in the application: main page, advanced search page, and result page. Users can upload a picture and select options in the first two pages, and show results in the result page.

Based on the above implementations, we successfully created an application with a fashion recommendation service.

References

[1] Jang, Y. R. (2018, July 6). Alibaba opens 'fashion AI store' at the year-end. TIN News. Retrieved from https://www.tinnews.co.kr/15395

[2] Hasegawa. (2017, December 28). The IoT revolution brought by Zozo town. Kotra News. Retrieved from https://news.kotra.or.kr/user/globalBbsDataAllView.do?dataIdx=163841

[3] Lee, S. W. (2018, June 8). Amazon launches 'Echo look' that checks fashion with artificial intelligence camera. Bodnara News. Retrieved from https://www.google.com/enKR885KR885oq=rnrmf+qjsduraqs

[4] KOTRA Stockholm Trade Center. (2018, June 5). Sweden opens the era of customized fashion just for you. Kotra News. Retrieved from http://www.kofoti.or.kr/textile/boardView.do?Code=TRENDUid=1041