# Visual Recognition

Haiyan Tian htian35@wisc.edu
Fengyi Zheng fzheng28@wisc.edu
Ziwei Zhao zzhao279@wisc.edu

## Abstract

*Visual Recognition has been a newly developed but powerful feature that could affect both individuals and businesses, especially for business like fashion industries that rely heavily on images and visual effects. As more fashion industries began to make their off-line to online retail transition, visual recognition began to assemble its importance. Nowadays, there are trends where consumers wish to find resembling products by uploading a photograph with a certain feature, in such cases, an accurate visual recognition tool is essential to simplifying the process. In this project, we wish to develop a a model similar to the visual recognition tool. We have three main goals: Use Convolutional Neural Network to extract features and classify images, apply Guided Backpropagation to understand the latent factors behind model and interpret the prediction, and leverage DCGAN to generate new clothes images. For the main model, we employed a ResNet152 model, pre-trained with ImageNet dataset. By training the model with 25 epochs, a batch size of 256 and a learning rate of 0.0001, attaching the Adams optimization, we were able to achieve a test accuracy of over 91 percent. And finally, successfully generated new images.*

## 1. Introduction

Due to the rapid progress of the society development, many people's expectations from clothing products have transitioned way past mere warmth providing, and into pretty visual effects. Such transition has moved people's attention more from the clothing product's quality, and into "keeping up with the latest fashion", resulting in them tending to find the products resembling a certain feature of interest while shopping. With more fashion industries began to explore ways to retail online, being able to recommend products based on an uploaded image are becoming more essential for the businesses that wish to become market monopolists. Thus, in such environment, images have began to gain its importance. Before developing tools like Convolutional Neural Network, several tryouts have been launched to achieve visual recognition and simplify searching process. For people, one can always gain information from images faster than texts. When people communicate, people would prefer communicating with plain images for convenience instead of describing what they want with textual attributes. And when people recall a certain piece of memory, it was always the images that are more vivid. In such circumstances, searching for images, and using images to conduct search are due to be made possible. This trend started in the late 1990s when AltoVista first launched an search engine that takes short text descriptions as input, and return images[1]. However, such feature relies more on image descriptions and cannot process images themselves. In 2001, the trending searches for Jennifer Lopez' green gown inspired Google Images to be the first to develop a search engine that takes images themselves as input, combine with the surrounding keywords, and return resembling pictures.[2] However, such search engine still relies on the indexes people use to describe an image. And when the person querying and the person writing the descriptions had different expressions, the search would hard to be exact. CBIR (Content-Based-Image-Retrieval) made analyzing the image contents possible. Images features like colors and shapes can be directly linked to another image without any text clarification. TinEye was the first to adapt "reverse" image search, enabling search engines to implement features that clustered similar pictures together in response to an image input. [3] After that, networks like CNN were gradually added to the search engine networks to make the searches more precise.

Figure 1. An image to illustrate similar clothings

Eliminating the process of manual feature extraction, CNN represents a huge breakthrough in image recognition. In the realm of image classification, facial recognition technology and image-to-image search have been the biggest advancements, and CNN can be found at the core of every process.[4] Especially after the launch of Google's Image Swirl project, CNN was used to develop search engines that allow users to explore similar images, which was adapted

to the "related images" feature today. However, while the famous ImageNet Dataset of Stanford illustrated the powerful usage of Deep Learning, the limitations began to reveal as most DL models are usually specialized to a particular domain or even a special task. Thus, Transfer Learning, as a solution to the losses caused in cross-domain tasks, has been developing and making its way into Deep Learning Network since 1995, the Neural Information Processing Systems(NIPS) workshop "Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems". [5] As transfer learning has proved to be a powerful way to put the knowledge from previously trained models into new areas, we decided to adapt the pre-trained models to enhance our accuracy after a few experiments.

In this project, we aim to use CNN and Back Propagation to extract features and classify images, and in the end, use Convolutional GAN to generate new iamges. After settling down the topics, we went through all the neural networks introduced in lectures and assigned readings, and chose to experiment with ResNet152, Inception and VGG16. Problems like image size mismatch and not enough CPU memories rose during the process, the problems were solved by resizing all images and transitioning to online Google Colab. After all the problems were solved, we tried the transfer learning technique by using the pre-trained models and compared the computed accuracy, finding that ResNet152 produces the best results. After developing the model, we applied back propagation to make classifying products more accurate, and used Deep Convolutional Gan to produce new images based on current image features.

The report is organized as follows: Section 2 introduces other similar projects done in the history, including brief introductions to their models. Section3 introduces the model used in this project, and the reason why we chose to use such network. Section 4 illustrates the experimenting process, including data preparation process, model training, and introductions to Back Propagation and DCGAN. The last few sections in this report will summarize our results, acknowledge the help we get, discussing the meaning and future potentials, and the group members' contributions.

## 2. Related Work

**The Google Image Swirl Project.** Google Image Swirl is an experimental search tool that organizes image-search results, by looking at pixel values to "understand" the image content, and organize and present them in visually distinctive groups. In the Google AI Blog presented by the Google engineers Y.Jing and H.Rowley, the process was elaborated: The Google Image Swirl takes both traditional texts and images as inputs, the engineers dig information from the image mainly by apply novel graph-analysis algorithm (The details of the model and the algorithm are classified and not shared). The engineers set all the (sub)classes before hand, and by applying algorithms, one can discover the high-order similarities to a certain category. In this project we adapted a similar method for data preparation, but worked with a pre-trained model and back propagation to recognize and classify the images.[6]

**Pulsar's Vertical AI.** Pulsar's Vertical Analysis is a identification tool that can help one to identify and categorize the text and/or visual content within a specific domain. This searching setup has been introduced as a tool to help provide a more accurate understanding of the searching results, as the results can be tagged and categorized within a specific subject. In August 2018, Pulsar introduced 7 popular searching modules, and several usages of their Vertical Analysis, including Emotion Analysis, General Image tagging, Colour Detection, etc. In the "TRAC: AI Modules" of Pulsar's website, it was introduced that functions like Emotion Analysis used machine learning to return confidence scores for different emotions, in order to detect and categorize a certain expression. And features like General Image Tagging adapted Deep Learning Algorithms, especially via DenseNet visualization (the details of the models are classified and not elaborated).[7] Pulsar's Image Analysis enables users to perform instant visual mining on images from social media, and analyse the content or get an understand of the subject of the image, as the searching results are ensured to be within the desired context. The project of Pulsar's Vertical AI is inspiring and similar to the purpose of our project. We too considered DenseNet for our training, but gave up as ResNet 152 gave a better and faster performance. In this project, we focused on the "fashion product detection module", the expectations of identifying products are similar to Pulsar's.

**The Pre-Trained ResNet152 Model.** In the paper presented by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, the statisticians tested various networks on the ImageNet 2012 classification dataset to compare their outcomes. The model was trained on the 1.28 million training images of 1000 classes, and evaluated on the 50k validation images. The performance of the models were evaluated by top-1 and top-5 error rates. Before the experiments, all the images were augmented on a scale randomly chosen from [256, 480], and for every image, only a random crop of 224*224 size is used.

After few comparisons, they switched from plain networks to residual networks with different layers, and soonly discovered that the 50/101/152-layer ResNets have a more accurate results than the 34-layer ResNets. They display a much smaller Top-1 (and/or Top-5) error, and the accuracy degradation problem hardly occurs as the depth increases. And overall the ResNets have a better performance than the VGG-16/19 while maintaining a lower complexity.[9]

Thus the ResNets became rather the optimal choice. In our project, before trying the pre-trained models, we tried training the VGG-16, Inception and ResNet-34/152, and eventually observed that the ResNet 152 has the best performance. After learning the benefits of the transfer learning, we switched to the pre-trained models as well.

## 3. Proposed Method

In this project, we mainly proposed three methods:

- ResNet152 pre-trained model to classify images.
- Back Propagation to interpret model and understand the main features extracted.
- DCGAN to generate new images using training set.

### 3.1. ResNet 152 Model

The first objective is to obtain a model that can classify images. There are three metrics that we take into considerations: accuracy, complexity, and repetitiveness. Two building blocks are essential to achieving our goal: architectures and training methods.

For the model's architecture, we decided to adopt Convoluntional Neural Network, more specifically, ResNet-152. Overall, ResNet's architectures allow researchers to implement very deep architectures due to its property of allowing skipping connections when all the weights and biases are zero. This can also help with the vanishing gradient problem. As it indicate in Figure 2 below, that if weight and bias goes to zero, then the activation value for the current layer will be the same as the activation value for the previous two layers. This is due to the identity function and RELU activation function. This property allows the network to expand in depth while reducing unnecessary complexities.

$$\begin{aligned} a^{(l+2)} &= \sigma\big(z^{(l+2)} + a^{(l)}\big) \\ &= \sigma\big(a^{(l+1)}W^{(l+2)} + b^{(l+2)} + a^{(l)}\big) \end{aligned}$$

Figure 2. ResNet shortcut

The ResNet-152 we implemented for our project used the properties of skipped connection as described above and Bottleneck design to achieve 152 layers. Just like the other ResNets, the ResNet152 adapted 7*7*64 dimension with stride 2 for the 1st layer and the 3*3 max pooling for the 2nd layer, as well as the average pooling layer after the 5th layer. Accordingly, ResNet-152 has 3, 8, 36, 3 blocks for the 2nd-5th convolutional layers, and has 11.3 billion FLOPS. [9]

Furthermore, to achieve higher performance with more stability, we leveraged transfer learning techniques. Transfer learning allows us to fine tone our data-set on a pre-trained model with large datasets. The advantages of the pre-trained model allow us to establish a more stable outcome. However, it also comes with disadvantages as the large dataset involves many images that has no association with our dataset.

Thirdly, to help the model learn faster, we decided to use Adaptive Learning Rates, specifically, ADAM. Overall adaptive learning rates will decrease learning if the gradient changes its direction, and increase learning if the gradient stays consistent. In the end, we employed ADAM as our stochastic optimization as it is able to incorporate the Momentum and RMSProp method. As shown in the Figures 3, 4 and 5, ADAM is a combination of those two methods:

$$m_t := \alpha \cdot m_{t-1} + (1-\alpha) \cdot \frac{\partial \mathcal{L}}{\partial w_{i,j}}(t)$$

Figure 3. Momentum-like term

$$r := \beta \cdot MeanSquare(w_{i,j}, t-1) + (1-\beta)\left(\frac{\partial \mathcal{L}}{\partial w_{i,j}(t)}\right)^2$$

Figure 4. RMSProp term

$$w_{i,j} := w_{i,j} - \eta \frac{m_t}{\sqrt{r} + \epsilon}$$

Figure 5. ADAM

All the procedures that we took allow us to construct a final model that can have qualities such as high accuracies, less complexities, and the ability to reproduce.

### 3.2. Guided Back-Propagation

Guided Back-propagation is a method that can help us to understand what the neurons do in a convolution network. This will help us to achieve our second objective: to interpret the model by sharing the same lens as it.

Guided Back-Propagation is an algorithm that can recursively search how neuron is affected by patches from the previous layer. This allows the researchers to see beyond the top layers and observe the effect of all the layers in the network.

To understand to mechanism behind Guided Back-Propagation, first we need to understand that the purpose of such method is to observe which pixels in the input affect the Neuron the most. So we are only interested in the features that can be detected by the neuron. In other words, we are performing gradient descent on the output and trying to maximize the likelihood of the output.

Similarly, during Guided Back-propagation, we were trying to maximize the input. In order to achieve that goal we set all the negative gradients to 0 as we do not care if a pixel doesn't have any positive effect on the neuron.

In our model, specifically, we want to use Guided Backpropagation to understand which input pixels has the largest impact.

### 3.3. DCGAN

At first, we considered adapting Convolutional Gan, however, due to our slight large image size (224*224 at this point), we were advised by professor Sebastian Raschka to adapt Deep Convolutional GAN to compete the constraints of ordinary GAN model. GAN is a generative model that competes against an adversary. The model mainly consists of two parts: Generator (G), which can extract main features from the images, combined with random noise, and generate new images. And Discriminator (D), which can distinguish generated fake images from the real ones. Compared to GAN, DCGAN made few changes such as eliminating fully-connected hidden layers in favor of a deeper network architecture, and utilizing batch normalization in different places for G an D. Training G, we aim to generate better quality fake images, to maximize the probability of D making a mistake. Meanwhile, we train D to get better at distinguishing images. The main theory of DCGAN is shown in Figure 6:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Figure 6. GAN's Main Theory

Almost all the activation functions are ReLU for G, and LeakyReLu for D. The functions in the last layers are Tanh and Sigmoid accordingly. For a better result, the learning rates are set at 0.0002 for both. For the optimizer, we used Adam for both to accelerate the training.

## 4. Experiments

### 4.1. Dataset

The dataset we trained our models on is the "Fashion Product Images Dataset" on Kaggle. The size of the dataset is over 30GB. It contains a images folder consisting images of over 40000 different fashion products, and the corresponding descriptions and image links summarized in two csv tables . See Figure 7 for the csv table containing the main descriptions, the other table is the json links matching all the image ids.

| id | gender | masterCategory | subCategory | articleType | baseColour | season | year | usage | productDisplayName |
|---|---|---|---|---|---|---|---|---|---|
| 15970 | Men | Apparel | Topwear | Shirts | Navy Blue | Fall | 2011 | Casual | Turtle Check Men Navy Blue Shirt |
| 39386 | Men | Apparel | Bottomwear | Jeans | Blue | Summer | 2012 | Casual | Peter England Men Party Blue Jeans |
| 59263 | Women | Accessories | Watches | Watches | Silver | Winter | 2016 | Casual | Titan Women Silver Watch |
| 21379 | Men | Apparel | Bottomwear | Track Pants | Black | Fall | 2011 | Casual | Manchester United Men Solid Black Track Pants |
| 53759 | Men | Apparel | Topwear | Tshirts | Grey | Summer | 2012 | Casual | Puma Men Grey T-shirt |
| 1855 | Men | Apparel | Topwear | Tshirts | Grey | Summer | 2011 | Casual | Inkfruit Mens Chain Reaction T-shirt |

Figure 7. Screenshot of the style.csv file

### 4.2. Data Gathering and Preprocessing

The data was downloaded straightly from Kaggle website. The dataset contains a folder of 44442 images of different fashion products, a csv file containing their descriptions (gender, MasterCategories, season, color, etc.), and a csv file containing their corresponding ids and URL links. The Data Preparation process mainly contains four parts: image resizing ,id combing, classifying, and zipping files.

During the project, we resized the images seveal times. The original images dimensions ranged from 3744*5616 to 150*200, which caused several size mismatch errors. We used the package "opencv" to resize all the images to the same 1080*1440 size, but based on the time it took to run an epoch, we calculated that it would take at least 9 days to run a model process. So we resized them to 128*128 to simplify the process, and eventually to 224*224 in order to use the pre-trained transfer learning models. For the last part of our project, we resized the images again to 64*64 for improve DCGAN's performance, and for efficiency. During training, we noticed there are id mismatch errors, after combing through all the csv files and the image ids, we noticed that there are 42227 ids with descriptions, and yet only 44442 images to relate. We went through all the ids and filtered out the empty ids before continuing the experiment processes.

After cleaning the data, we listed the frequencies of all the categories and the befitting gender of fashion products, and decided to keep the 2 master categories that has the most frequencies: "Appeal" and "Accessories", and listed the rest as "Other". For the gender, we combined "Boys" with "Man, and "Girls" with "Women". We defined them as class 0 - 5 for the 6 different combinations. (See Figure 3 for the relative frequencies of each class)

Class 0: Men/Accessories Class 1: Women/Accessories.

Class 2: Men/Appeal. Class 3: Women/Appeal.

Class 4: Men/Other. Class 5: Women/Other.

After classifying all the images with their ids, we shuffled all the data and divided it into training set, validation set and testing set on a 75:10:15 ratio. With the adjusted image size, the model still takes hours to run and the group's laptops often encounter force quits because of heating and lack of memory. So we uploaded everything to Google Colab, zipped and unzipped the datasets, and access through shareable links on cloud to quicken the process.

### 4.3. ResNet 152 Model

In the model selecting process, we tested with several models before deciding on ResNet, mainly with VGG16 and ResNet152. First, we randomly selected 1000 images from the original data-set and split them into Training, Validation, and Testing sets on the same ratio as before. Then we applied the pre-trained models of VGG16 and ResNet152 from Pytorch on the selected dataset to test

which model has a better performance. The end result shows VGG16 provided a slightly higher performance than ResNet152, but both have accuracy under 80 percent. We eventually adapted the pre-trained ResNet152 model for the final data-set as the ResNets has it is able to establish depth without more complexity.

As for hyper-parameters, we used a learning rate of 0.0001, a batch size of 128 and 25 epochs to train. We trained our model on the final two layers of the pre-trained model from Pytorch and kept the other layer as default. We decided to stop our model at 25 epochs because it seems like without at least double the number of the existing epochs, the validation accuracy will most likely saturated around 91 percent. So, as we weighted time against the small increase in accuracy and decided 25 epochs for now is sufficient. To evaluate our model, we simply use the accuracy rate. On average, it takes a bit under one and half hours to train with about five minutes per epoch.

### 4.4. Guided Back-propagation

Guided Back-propagation was only performed on the testing set to understand the characteristic the model extract for each classes. First, we needed to unnormalize the images. Then we performed guided back-propagation to see how the feature was extracted for each classes with correctly predicted labels and for classes with incorrectly predicted labels. However, the observation above is based on one image, whereas our goal is to understand the main feature for classes overall. So we performed an average guided back-propagation on all the classes by adding the gray-scale pixels for one predicted class then dividing it by the number of predicted image in that class. Then we repeated the above procedure for all the classes.

### 4.5. Deep Convolutional GAN

After successfully extracting main features from the images, our group decided to take the project a little further by adapting DCGAN model, to generate new pictures. In reality, some consumers may want the system to recommend new but similar styles if they are exploring their preferences.

For efficiency purposes, we resized all the images to a dimension of 64*64, with 3 colour channels. We only tested with the training dataset as it is the largest of the 3 sets, before testing the model, all the images were normalized so that the pixel values range from -1 to 1. We set the batch size to be 128, and the learning rates at 0.0002. While testing the model with 100 epochs, our laptops broke down because of memory constraints, in such circumstances, we reduced the number of epochs to 20, which may affect the results as the quality should be increasing with more training epochs, but efficient at saving time and resources.

However, we figured that in order to maximize the model's utility, it's better to test the DCGAN model on a specific class, as in reality most people would wish the system to recommend products of the same category. Revisiting the 6 classes we categorized before, we divided the dataset into 6 small sets, and tested with the DCGAN separately to see the generated results. The results can be seen at the Results section.

### 4.6. Software and Hardware

All the project was done on the group members' laptops. All the code will be submitted online along with the project report.

All the coding was done on Jupyter notebook, R and mainly Google Colab. R and Python 3.7 were mainly used in the initial state of the project when we prepare the data and test the time it took to run the model. After we discovered the damage the process does to the laptops we switched to Google Colab, using the linked GPU to quicken the process and share the code in time.

## 5. Results and Discussion

### 5.1. ResNet 152 Model

After running 25 epochs, we could achieve a training accuracy of 93.26 percent, a validation accuracy of 91.26 percent, and a test accuracy of 91.75 percent. As shown in Figure 8, there is no obvious over-fitting as the difference between train and test accuracy is less than 2 percent. Another thing to notice is that our test accuracy performs slightly better than our validation accuracy. This is not a common result, which could means our model is robust overall. So, we decided dropout is unnecessary.


Figure 8. The Accuracy Results

Furthermore, we want to understand if there any biases in our model. More specificity, will more data in one class have any effect on the overall predication of our model. In the Figure 9 and 10 below, we compared the training dataset's distribution with the predicted label distribution. These two distributions bears high resemblance. This is a good

thing because it shows our model does not have biases towards any classes label despite the fact that some classes have more data. However, our model can only has a test accuracy of 91.75 percent. So, some later work could be investigating the distribution of incorrectly predicted labels to get a more complete look of our model.

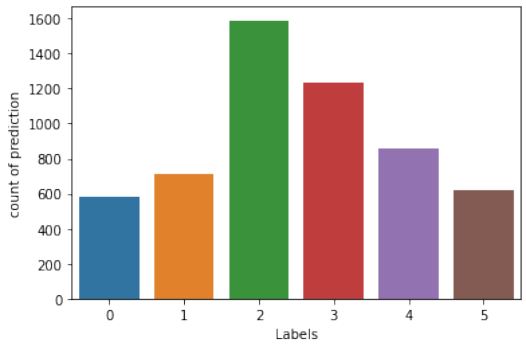


Figure 9. predicted distribution

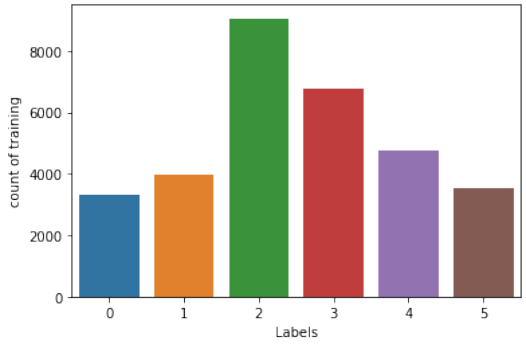


Figure 10. training distribution

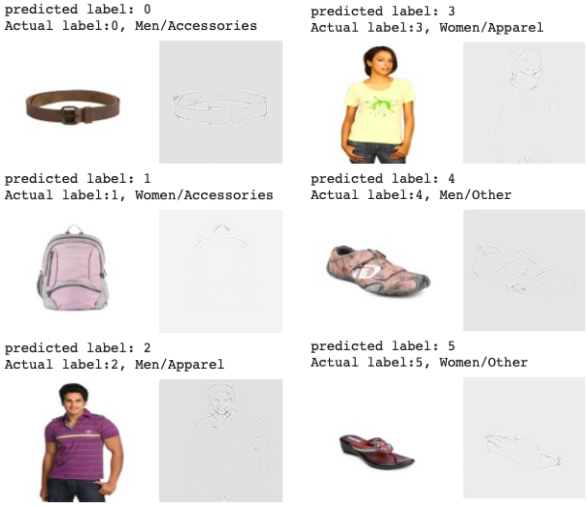


Figure 11. guided back-propagation


Figure 12. average guided back-propagation for predicted label 0

## 5.2. Guided Back-propagation

We first explored guided back-propagation on individual images for all six classes. In Figure 11, for predicted label 0, the guided back-propagation just shows an outline of the original image. While for predicted label 1, the backpack, the guided back-propagation picked up on the handle part instead of the entire outline. For label 2 and 3, the guided back-propagation picked up on model's faces as opposed to actual clothing. Similarly to predicted label 0, predicted label 4 and 5 picked on the overall outline. However, individual images can only tell us limited information on what feature our model extracted. So, we used average guided back-propagation to gain a more holistic view of our model. A subtle change is that the pixels captured by the average guided back-propagation are a lot darker which makes it easier to see the feature extracted compared to the individual images guided back-propagation.

It is worth noticing that the average guided back-propagation performed on male accessories resulted in a pair of glasses. It's possible that there are more glasses images in this class in our dataset.

For label 1, unlike the individual back-propagation we observed earlier, the guided back-propagation shows the lower part of a hand bag. This could mean there are more hand bags in our dataset. For predicted label 2, the result is similar to the individual back-propagation we observed earlier that instead picking up details on clothing, the our model extracted face as the more important features. This observation is the exact opposite for predicted label 3. As opposed the individual back-propagation that model picked up on the face, the predicted label 3 actually extracted an almost t-shirt like object as the most important feature. It could mean that there are less images with faces in predicted label 3.

Label 4 show a shoe resembling to a sneaker and label 5


Figure 13. average guided back-propagation for predicted label 1

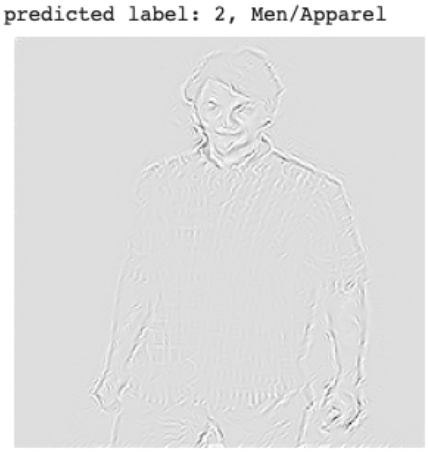


Figure 14. average guided back-propagation for predicted label 2


Figure 15. average guided back-propagation for predicted label 3

shows a sandal. It could mean that there are more sneaker images for men and sandal images for women. Also for label 4 and 5, there are personal good images as well. But there is no sign of them in the images, which could mean


Figure 16. average guided back-propagation for predicted label 4

the sample size is too small to be significant.

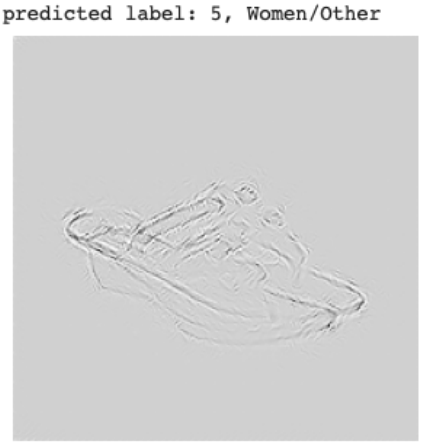


Figure 17. average guided back-propagation for predicted label 5

### 5.3. Deep Convolutional Gan

By using the DCGAN, we produced new images both on the whole training set, and on all 6 individual classes.

From the results we can see that the fake images are relatively similar to the ones in the dataset in three aspects: colour, styles and figures. In the original dataset, the fashion products are mainly colourful classic styles of clothing and shoes, occasionally appearing with the model's image. The colours of the products can be distinguished easily, matching the colourful images of the original dataset. From the generated fake images, we can distinguish the rough outlines of the products, mostly basic style products like the original images, occasionally with the figure of the model. However, there are some images that are a little blurry and may not be distinguished clearly, and there are 2 failures (row 5, column 5 and 8) that cannot be distinguished at all. The image is all blurry and we cannot recognize a figure.

We believe that a more complex model may be used to generate a more clear outcome, or training with more epochs with different hyperparameters.

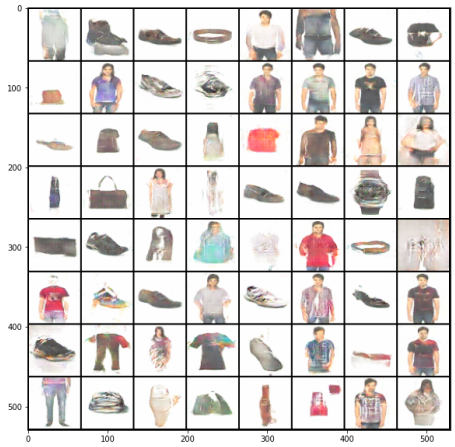

Figure 18. 64 New Images Generated by using the whole Training Set

After testing with the whole training set, we tested with all individual classes. In Figure 19 are the results tested with class 1 and 2 data. The results are pretty promising as the figure of the product can be distinguished, and most images are of the same category. However, there are few failures in the process, like when we are generating using class 0, the image results are all blurry and cannot be distinguished, probably due to the small size of the dataset. (the relative frequency of each class can be found in Figure 3)

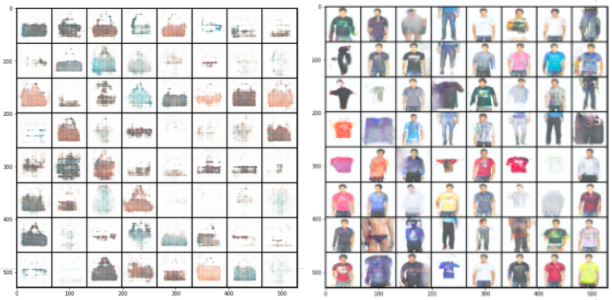

Figure 19. New Images Generated Using Class 1 data (left) and Class 2 data (right)

Figure 20 illustrats the generator and discriminator losses from our experiment. The losses are competing each other as expected, but are still noisy and oscillating, it's possible that it is because of limited training epochs, and hyperparameters that can be improved in the future.

### 5.4. General Pros and Cons Discussion

An advantage of our project is that, we chose the model ResNet152 after a few comparisons. Before settling down on the final decision, the group separately tested different

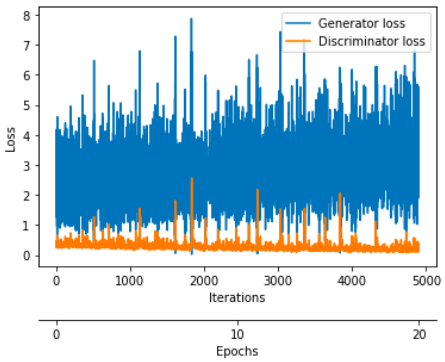


Figure 20. Generator and Discriminator Loss

models (both directly and pre-trained) on their laptops, and compared the performances. The tryouts included models like VGG16, DenseNet, and Inception, and ResNet152 achieved the highest accuracy. The high accuracy results benefited from the model selection process.

A disadvantage of the project is the affected quality of the images. The images were resized several times during the whole process, for purposes of better fitting the model and increasing efficiency. The program was operated on the same set of images, during the resizing process, there are images being "shrink" and becoming blurry, thus losing some information. This process is likely affecting our final results.

## 6. Conclusions / Future Directions

The future improvements can be discussed in three sections: Back Propagation, Model Training, and DCGAN.

Due to the diversity of the original images, it became difficult for the Back Propagation model (refer as BP) to distinguish image features. The images in the dataset contain various styles: some may only contain the product image, while others exhibit a model wearing it. In such circumstances, we noticed that there are few times when BP recognized the model's face instead of the product as the target. In future, we will try to eliminate the useless information of the image, to improve the BP's performance. Also the results produced by BP are grey-scale while the original images are color-scale, in future we can try to find an algorithm that can produce color-scale images.

In the project we adapted the pre-trained model ResNet152, bacause the model was pre-trained on ImageNet dataset, it might not be the perfect fit for our relatively small Fashion Dataset. We can train the model better to fit our dataset. For the DCGAN, due to the time constraints, there might be better hyperparameter values that we haven't discovered. So, there are still rooms for improvement.

## 7. Acknowledgements

The authors would like to thank Professor Sebastian Raschka for his helpful feedback. During the experimenting process, the authors came across another project posted on Kaggle which used the same dataset.[10] The introductions inspired the authors other ways to adjust the images and prepare the dataset better. The authors would like to thank Marlesson for his introductions.

## 8. Contributions

The authors worked synchronously on different laptops and communicated through WeChat video call in time to discuss the process and the newly risen problems. During the data-gathering part, each group member contributed a large amount of time searching through different online platforms like Kaggle to gather useful datasets, and did research to find the most befitting network to develop. Haiyan was mainly responsible for cleaning the data, adjusting the size of the images and cleaning out mismatched ids. Fengyi was mainly responsible for developing Guided Back-propagation and wrote about all the sections involves Model and Guided Back-propagation in the report and presentation. Ziwei were mainly responsible for employing the DCGAN model. During the experiments, the group members experimented different models on their laptops separately at the same time, and evaluated the results together. All group members took a heavy role in researching and developing the project report.

## References

[1] WordStream."The History or Search Engines" https://www.wordstream.com/articles/internet-search-engines-history

[2] R.Tashjian. Sep 20, 2019. "How Jennifer Lopez's Versace Dress Created Google Images". https://www.gq.com/story/jennifer-lopez-versace-google-images

[3] I.Pinto. March 21, 2018. "A Brief History of Image. Search"https://www.syte.ai/blog/brief-history-image-search/

[4] A.Bonner, "The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification". https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb

[5] D.Sarkar. Nov 14, 2018. "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning". https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

[6] Y.Jing, H.Rowley, Google Research. November 23, 2009. "Explore Images with Google Image Swirl" https://ai.googleblog.com/2009/11/explore-images-with-google-image-swirl.html

[7] L.Maruta, "TRAC: AI Modules", April, 2020 (updated). "https://intercom.help/pulsar/en/articles/520951-trac-ai-modules"

[8]

[9] K.He, X.Zhang, S.Ren, J.Sun. December, 10, 2015. "Deep Residual Learning for Image Recognition"

[10] https://www.kaggle.com/marlesson/building-a-recommendation-system-using-cnn-v2