CS230

Fashion products synthetic image generator (Generative Adversarial Network)

Jinchuan Shi
Department of Computer Science
Stanford University
jinchuan@stanford.edu

1 Introduction

We are going to develop a deep neural network that can do the following two tasks. First, it takes an image as input and then automatically labels the attributes of it, this request classifies the image into multiple feature classes, each feature class corresponding to one of the attributes of the image. Second, the network can take a list of the attributes and generate a synthetic image that satisfies all the requested attributes. We believe such a neural network has a deep understanding of the image's attributes, and also it can apply to the image data argument process to help the training with generated synthetic data. Also, with transfer learning, this kind of neural network can be used to generate other kinds of synthetic data besides the images, it will be very useful for tasks when getting real data is difficult.

2 Dataset

We use the Fashion product images dataset small from the Kaggle dataset collection. The data set contains around 44k fashion product images, each of the image is 80 x 60 x 3 dimension. There is a style excel file that describes the attributes of all the images. For each image, there is a total of 10 attributes, Figure 1a shows the statistic of this file. For this project, the multi-label classification part, we pick the attributes sub-category and the gender as our classification targets, the statistic of those two column shows in Figure 1b and 1c.

First, we load all the images to a data frame X; the target dimension is 80 * 60 * 3. The shape of X is (44440, 80, 60, 3); this is our model's input. Second, we apply the one-hot encoding on the sub-category column and gender column, add the one-hot columns to the CSV file, then drop all other original columns. After that, we create the reference data frame Y, the shape for Y is (44440, 25), and this is our multi-label classification output. We pick the top 20 attributes in the sub-category based on the attributes count and all 5 attributes for the gender. Finally, we split X and Y into the training sets and validation sets with a ratio of 19 to 1 and resulting in 42218 data points for training and 2222 data points for validation. We don't have the test set for this model. For the second part, the variational autoencoders model, we reuse the same training set and validation set.

3 Methods

3.1 The multi-label classification neural network

For the image multi-label classification part, our method is using one of the classic convolutional neural networks [5] [1] [2] and then apply transfer learning to it. The base model we choose is the

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

```
RangeIndex: 44440 entries, 0 to 44439
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  44440 non-null  int64
 1   gender              44440 non-null  object
 2   masterCategory      44440 non-null  object
 3   subCategory         44440 non-null  object
 4   articleType         44440 non-null  object
 5   baseColour          44425 non-null  object
 6   season              44419 non-null  object
 7   year                44439 non-null  float64
 8   usage               44123 non-null  object
 9   productDisplayName  44433 non-null  object
 10  image               44440 non-null  object
dtypes: float64(1), int64(1), object(9)
```

(a) Fashion images style header information

```
Topwear                    15401
Shoes                       7344
Bags                        3055
Bottomwear                  2693
Watches                     2541
Innerwear                   1808
Jewellery                   1080
Eyewear                     1073
Fragrance                   1012
Sandal                       963
Wallets                      933
Flip Flops                   915
Belts                        811
Socks                        698
Lips                         527
Dress                        478
Loungewear and Nightwear     470
Saree                        427
Nails                        329
Makeup                       307
```

(b) Sub-category statistics

```
Men        22160
Women      18631
Unisex      2164
Boys         830
Girls        655
Name: gender, dtype: int64
```

(c) Gender statistics

MobileNetV2 [9]. For the transfer learning, we removed the top layer of the MobileNetV2, then added a global average pooling 2D layer to the output of the MobileNetV2.

Then we add two dense layers, the first one has an output of 1024, and the activation function is relu, the last dense layer is the output layer that has an output of 25, and the activation function is the sigmoid function. The dimension 25 is the dimension of the one-hot encoding for the sub-catelogy and gender attributes. Since we are doing multi-label classification, we need to use the sigmoid function instead of the softmax function for the output layer. While the training, we freezed all the pre-trained weights on the MobileNetV2, and only train the 3 new layers that we added on the top. The optimizer algorithm is using the adam, the cost function is binary crossentropy, and the metric we used is the Area Under The Curve (AUC). The cost function and the metric are adjusted for the multi-labels classification tasks.

3.2 The variational autoencoders

For the generator [10] [7] [6] [3], we use the variational autoencoders [4] as our baseline model, then we explored different encoder architectures. The main reason for choosing this neural network is this it is a probabilistic autoencoder, so the generated image could have some variations. Most importantly, it is also a generative autoencoder, meaning that it can generate new instances that look like they were sampled from the training sets, and this is exactly what we want for this project, to generate random synthetic fashion product images.

The original variational autoencoders, the network has two parts, the encoder network, and the decoder network. The encoder network has an input layer, then followed by two dense layers with the "selu" activation function, then followed by a Gaussian noise sampling layer. The Gaussian noise sampling layer output the random latent classes from the Normal distribution with the inputs' mean and standard deviation. The decoder neural network takes the encoder's output, the latent space coding as the input, and then followed two dense layers with "selu" activation function, the output layer will has the same dimension as the encoder input, which is the same dimension of the image. The original autoencoders are tested on the fashion MNIST images dataset, therefore, the input and output dimension is both (28, 28).

We did a serial neural network architecture explore based on the base model. To be specially, we tried the following variations:

- The first thing we did is to change the input and output dimension to match out new fashion image dataset, which are images with colors. The new input and output demension is (80, 60, 3), which is 18 bigger than the fashion MNIST images. This suggested our task is more complex than the original model designed for.
- The second change is we added the multi-label classification results to the latent space. Recall our multi-label classification model has an output with 25 dimensions, one-hot encoding represent the two attributes, the sub-catelgory and the gender. Our encoder will take the same image input, let the classification model to make predictions, then concatenate the prediction with the encoder's latent spaces. We fixed the weights in the prediction model and don't train it when train the autoencoder/decoder network.

- The third thing we tried is to increase the model's complexity, since our project tasks are more complex than the original model design and tested for. We decided to add several "convolutional auto-encoder blocks" and "convolutional auto-decoder blocks".
  - Each convolutional auto-encoder block contains three layers, it started with the Batch-Normalization layer, followed by a Conv2D layers, then a MaxPooling2D layers. All the Conv2D layer are using kernal size = 3 and activation function is "relu", but the number of the filter are different. All the maxPooling2D layers are the same, using pool size (3, 3).
  - Each convolutional auto-decoder block contains 3 layers, it started with the BatchNormalization layer, followed by a Conv2DTranspose layer, then followed by a Conv2D layer, we took this idea from the U-net [8]. All the Conv2DTranspose layer are using kernal size = 3 and padding set to "same" but the number of the filter are different. All the Conv2D layer are using kernal size = 3 and activation function is "relu", but the number of the filter are different.

  We tried to add different combination of the Conv-Encoder Blocks and Conv-Decoder Blocks with different filter configurations. After serial iterations, the best architecture is add 4 Conv-Encoder with filter size 8, 16, 32, 64, and don't add the Conv-Decoder.
- The final thing we iterated is the size of the latent space. we tried dimension 100, 50, 20, and 10. find out 100 is the best amount all the test cases.

This led to our final architecture of the neural network: The input layer (80, 60, 3) was followed by 4 Conv-encoder blocks with filters 8, 16, 32, and 64, then followed by two dense layers has 150 neural and 100 neural with "selu" activation function, then is the output the latent space coding with (100 dimensions) and concatenate with the multi-label production (25 dimensions), and this is the output layer of the encoder. Then the decoder (the generator) takes the output of the encoder, followed by two dense layers (100 neutrals and 150 neutrals) with "selu" activation function, then the output layer with dimensions (80, 60, 3) with "sigmoid" activation function.

The loss function for the generator is composed of two parts. The first part is the usual reconstruction loss that pushes the autoencoder to reproduce its inputs, we use the binary cross-entropy as the cost function. The second part is the latent loss that pushes the autoencoder to have codings that look as though they were sampled from a simple Gaussian distribution, and we use the KL divergence between the target distribution and the actual distribution of the codings. The optimization algorithm uses the "rmsprop" and the metric we are using is the rounded accuracy.

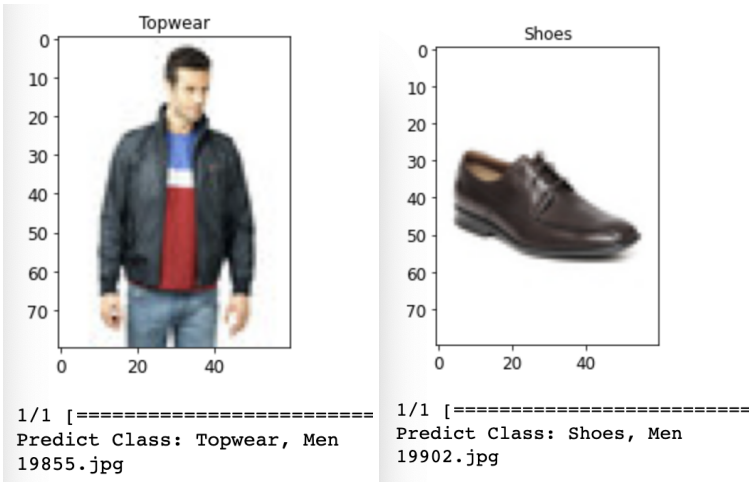
4 Evaluation

For the multi-labels predictions, After 10 epochs, the multi-labels production model can achieve a training error of 0.0120, and the AUC is 0.9993. For the validation set, the error is 0.0712, and the AUC is 0.9804. We can see from the testing images shows in the Figure 2, the predictions are all corrected.

For the image generation neural network, the variational autoencoders, we can achieve loss = 0.2105 and rounded accuracy = 0.9471 for the training set, loss = 0.2121, and rounded accuracy = 0.949 for the validation set. The results definitely can be improved, but due to the time and computation resources, these are the best results we can get for this project.

We did two synthetic image generation tests, the first one was the image reconstruction. We took the image as the input, then we use the variational autoencoders to make predictions, and the auto-decoder did the image reconstruction job. As we can see, the Figure 3, shows the reconstructed images, they are human recognizable.

The second test uses randomly generated seeds as the latent space codings, and concatinated with the user-defined sub-catelogy and gender, to generate the synthetic fashion images. For this test, we only used the auto-decoder (the generator) part of the network. As we see, the synthetic image quality is worse compared to the reconstructed images. We believe the main reason is for the reconstructed image, the latent space codings are not random, they are the output of the encoder network, so they contain useful information. On the other hand, the randomly generated latent space codings are pure noise and do not has any useful information, therefore, the generated image's quality is low. But some images, shown in Figure 4 are still human-recognizable.



(a) Man Topwear

(b) Man Shoes

(c) Women Bag

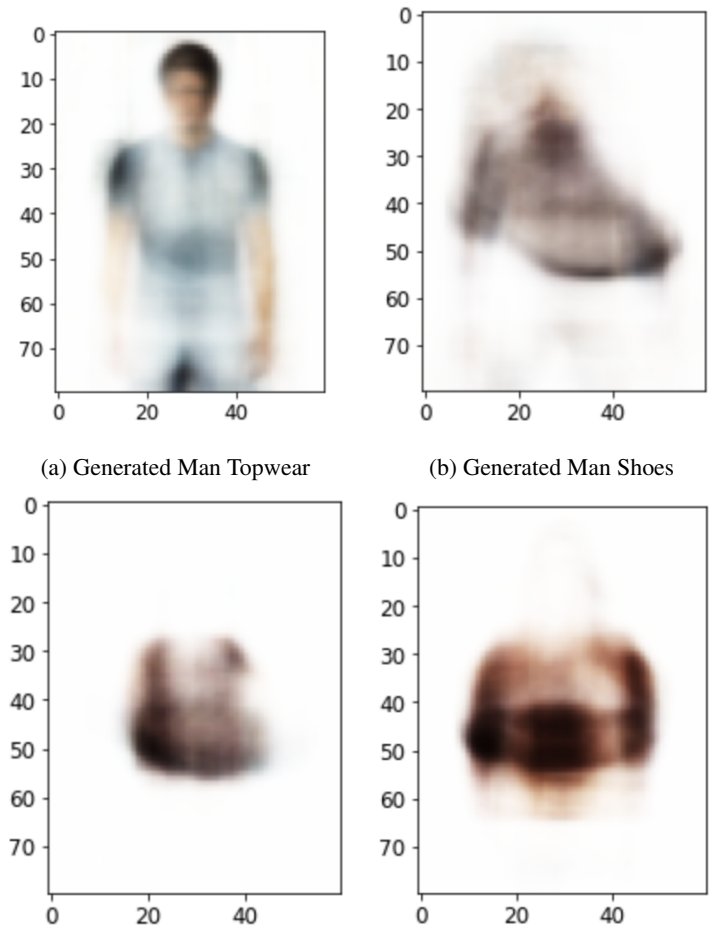
(d) Women Watch

Figure 2: For the multi-labels predictions, we can see from all the testing images (a), (b), (c) and (d) the predictions are all corrected.

Figure 3: Reconstructed Image from the auto encoder

5 Novelty and Contributions

Connecting the multi-label classification outputs with the variational autoencoder's latent space coding, then using the classified labels as the generator input to generate the synthetic on the fashion



(a) Generated Man Topwear

(b) Generated Man Shoes

(c) Generated Women Bag

(d) Generated Women Watch

Figure 4: Generated Synthetic Images using random seeds and assigned attributess

image data set is something new. We also did several neural network explorations to improve the performances.

In terms of contributions, my team only has one person. I did all the work including literature review, topic selection, dataset preprocess, implementing the training code, testing, making the video and writing the report.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[3] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.