AI FASHION

TEAM: Aaliyah Hänni, Dwight Sablan, Liem Luong, Vanessa Hsu
SPONSOR: Tatiana Teppoeva | Microsoft Sr. Data & Applied Scientist

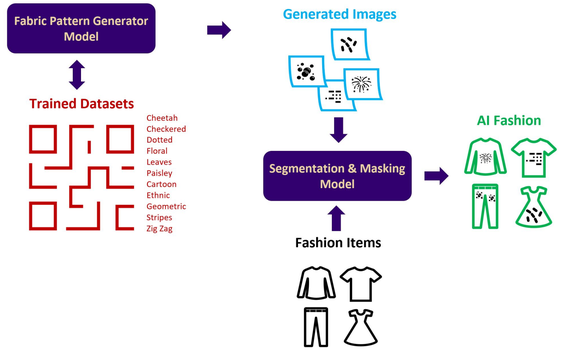
INTRODUCTION

The intersection of art and data science has gained more popularity in recent years. The applications between art and AI serve as inspiration for people to produce new works of art that can be more accessible to the broader audience.

In the fashion industry there is constant demand for new and interesting design patterns, which can often be expensive and time consuming to create. Fashion designers are bound by their own internal biases and face limitations on the level of creativity and in-demand fashion that they can create within a given timeframe.

With advances in deep learning, neural networks can be used to alleviate some problems in this field by improving the level of creativity in design for both quantity and quality outputs. In this work, we implement a neural network model to generate new textile patterns for many fashion items. Our solutions and findings could be easily used by the non-tech savvy and will make designs more accessible for the broader audience.

ARCHITECTURE



ARCHITECTURE DIAGRAM OF AI FASHION APPLICATION

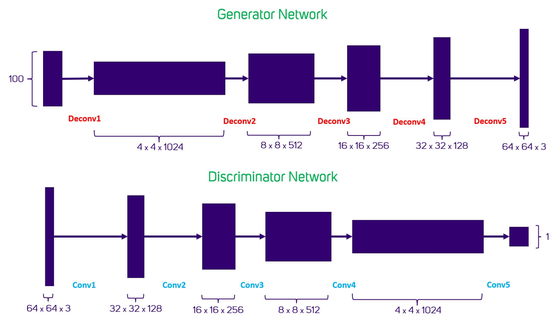
Our core architecture design includes two main components: Fabric Pattern Generator (component 1) and Segmentation & Masking (component 2).

In component 1, we apply the Deep Convolutional Generative Adversarial Network (DCGAN) by training with various textile pattern datasets. The output of this component 1 will be a neural network generated image input for component 2. In component 2, another pre-trained neural network model will perform image segmentation of clothing patterns and mask the new pattern over the fashion items to produce a result.

In this project, we use PyTorch to build models and Django to develop website. The final product is hosted on Amazon AWS

MODEL

DCGAN Model is the architecture being used for fabric pattern generation in this project. The advantage of DCGAN is its continuous improvement between the Generator and Discriminator Networks by improving the performance while reducing the loss of both networks to produce the best fake images. Below is the neural network structure of the Generator Network and Discriminator Network



DCGAN MODEL

Improving the model performance and reducing the loss are our objectives. We use the default Adam optimizer and Binary Cross Entropy loss from PyTorch.

Loss Functions and Optimizers

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)]$$

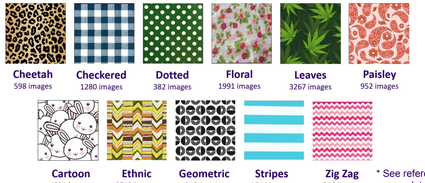
Above is the Binary Cross Entropy loss function for both *log(D(x))* and *log(1 - D(G(z)))*

- Objective of training the discriminator: **maximize the *log(D(x)) + log(1 – D(G(z)))*** with the aim to increase the probability of correctly classifying of input image as real or fake images.
- Objective of training the generator: **minimize the *log(1 – D(G(z)))*** with the aim to generate better fake image outputs.
- *D(x)* is the average output (across the batch) of the discriminator for the all-real batch. In theory, this number should start close to 1 and then converge to 0.5 when G gets better
- *D(G(z))* is the average discriminator outputs for the all-fake batch. In theory, this number should start near 0 and converge to 0.5 as G gets better

With the component 2 (Segmentation & Masking), we leverage the pre-trained model (Unet_2020-10-30). This model weights for clothe segmentation were trained over the Kaggle dataset - iMaterialist (Fashion) 2019 at FGVC6.

DATA

The training datasets for the DCGAN model are segregated into one of the 11 categories. These images are cleaned and standardized to the same height and width (224 x 224 pixels)

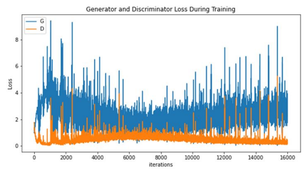


* See reference for detail of data sources

RESULTS

Fabric Pattern Generator Model: Among the six-pattern datasets we trained, we achieved good results for Floral and Cheetah of the fake images and lower loss values. Below is our floral result in a 4x4 matrix from the model.

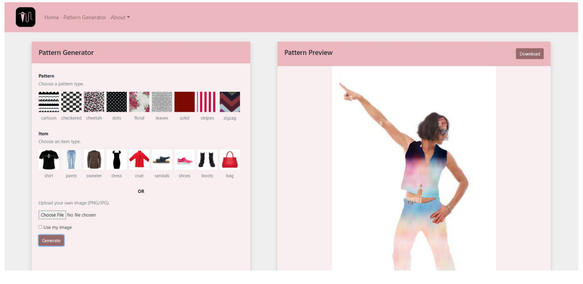
Learning rate:	0.00002
Batch Size:	128
Image Size:	64
Number of Channel:	3
Size of z latent vector:	10
Feature maps in G and D:	64
Adam Optimizer:	0.0005
Epoch:	1000



Below is sample of the output result after masking generated pattern (component 1) on to a fashion item (component 2)

WEBSITE

Demo: tny.im/CP0Ke



CONCLUSIONS

Among the eleven categories of patterns, we achieved best results – where the fake images generated from the neural network were closer to the real images - with floral, cheetah, paisley, and leaves. The take-away from the result is that DCGAN model works well for the type of images that does not possess a strict geometric pattern.

The other interesting observation is that we can improve the model by training with various settings. This will reduce the loss performance between Generator Network and Discriminator Network :

- A small Adam optimizer beta value (< 0.05)
- Smaller size of z latent vector (< 100)
- Learning rate between 0.0002 and 0.00002
- The larger training epoch, the better result it yield

The generated image from our model at this stage can be viewed as abstract art. There are no boundaries to how art can be produced and perceived as art can be subjective and influenced by a creator or viewer's baises.

LIMITATIONS

One of the limitations we experienced is that computer vision requires computing power – especially GPU – in order to train the neural networks. We gained better results with larger training epochs but it was very time consuming..

Second limitation is the limit on free available GPU resources from Google Colab we have.

Third limitation is the accuracy of result from component 2 based on various factors: image characteristic (transparency of the background, plain icon), and blend image with people from obscure view.

FUTURE WORK

Improve the model performance by using a better-quality datasets as well as altering the neural network layers to improve the accuracy of fake images.

Improve the image resolution from component 1 model output is a nice feature to add for this project.

Improve the performance of component 2 so that it can mask the graphic well for various scenarios from the above limitations.

REFERENCES

Model Sources

- Iglovikov, Vladimir. "Ternaus/cloths_segmentation: Code for Binary Segmentation of Cloths." *Code for Binary Segmentation of Various Cloths*, GitHub, https://github.com/ternaus/cloths_segmentation.
- "DCGAN Tutorial." *DCGAN Tutorial - PyTorch Tutorials 1.10.1+cu102 Documentation*, PyTorch, https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.

Data Sources

- lstearns86, "Lstearns86/Clothing-Pattern-Dataset: A Large Dataset Containing Images of Clothing Patterns." Clothing Pattern Dataset, GitHub, https://github.com/lstearns86/clothing-pattern-dataset.
- Additional patterns from web-scraping the following sites:
  - Fashion Fabrics Club, https://www.fashionfabricsclub.com
  - Online Fabric Store, https://www.onlinefabricstore.com