Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

School of Electronic Engineering

AI-assisted fashion design with deep generative adversatial networks

Project Portfolio

Saverio Pulizzi
ID Number: 18210293

August 2020

MEng in Electronic and Computer Engineering

Supervised by Dr. K. McGuinness

Acknowledgements

I would like to thanks Dr. Kevin McGuinness for giving me the opportunity to work on this interesting project and for his guidance.

Declaration

I hereby declare that, except where otherwise indicated, this document is entirely my own work and has not been submitted in whole or in part to any other university.

Signed: Date: 12/08/2020

AI-assisted fashion design with deep generative adversarial networks

Saverio Pulizzi

***Abstract*—Can fashion designers rely on a 2D web application to generate new design of fashion garments? In this research project, a state-of-the-art generative model has been deployed on a live web environment, where users are able to generate brand new fashion design images based on user provided handbags or shirts shape and preferred textures accessible at http://schedio.pythonanywhere.com/. The project provides first an overview on existent conditional generative adversarial network solutions applied to fashion design and then deep dive into the technology used and the technical implementation needed to build a web application. Challenges and limitations will also be explained and the results obtained from a users survey will show some of the challenges that will be addressed with proposed solutions to improve the user experience as well as the output quality of the generated image are reported.**

***Index Terms*—generative adversarial networks, web application, fashion design, AI system**

I. Introduction

NOWADAYS, the fashion industry is shaped by fast changing consumers preferences. In this fast-paced environment, fashion designers need to perform scalable and quick experimentation of their new design ideas. However, bulky, time consuming and sophisticated software represent the state-of-the-art technology used by fashion designers to create new design prototypes. Traditionally, the design process relies on a pattern maker who is responsible for the selection of the texture (together with the material and the fabric) and a sample maker who works mainly on the shape of the product [8]. A fast and reliable 2D solution to fashion design prototyping does not exist yet, while significant time and human capital investments are made by fashion companies towards the usage of existent garments modelling software. What if fashion designers could accelerate and augment their prototyping process with the aid of an artificial intelligence system able to generate new fashion design images? Imagine you could create hundreds of different combinations of colors and textures of your fashion sketch by simply uploading a sketch and a texture. In few seconds, you would be able to create different styles for your sketch and select the ones that aesthetically match with your tastes. The existing solutions to this problem are mainly 3D based while 2D solutions are just emerging. On the one hand, 3D solutions allow a better design customization but those are bulky, sophisticated and slow. On the other hand, the existent 2D solutions lack of a user interface and their output quality is often scarce. In order to speed up fashion design visualization and prototyping, the goal of this paper is to develop a web application that can generate a fashion design image by automatically combining a selected texture with a picture of a fashion sketch. The output will be generated by leveraging existing machine learning models that have been proven successful in preserving structural coherence. Those models are able to preserve the shape of the uploaded fashion sketch while matching it with the selected texture conforming to the uploaded image outline. This research project has proven successful in deploying an existing machine learning model into a real time web application which is able to provide, in few seconds, satisfying visual results of fashion design images to the final user. Indeed, the results coming from a user survey showed that the output quality is of a sufficient quality. However, future development could focus on improving the user experience and the resolution of the output image.

The author would like to thank the project supervisor Dr. K. McGuinness who is with the school of Electronic Engineering, Dublin City University, Dublin, Ireland

II. Review and analysis of prior work

Generating a fashion design image with a fabric pattern and a sketch as digital input is a problem of image to image translation. There are many existing researches about conditional generative adversarial networks applied to the fashion industry but only few practical implementations of those can be found.

From a research point of view, FashionGAN[6] and TextureGAN[11] are among the existing generative models for fashion image generations conditioned on textures able to generate a relatively good quality output given a sketch and a texture image as an input. However, while TextureGAN is able to produce acceptable results for any type of fabric pattern, one of the major weakness of FashionGAN is that irregular fabric patterns are not mapped correctly on the garment shape and, in such conditions, this cause the generation of very low quality of outputs. For this reason, TextureGAN is the model used in this project to perform real-time fashion design image generation.

Besides offline fashion design image generation, there have been very few attempts of building practical implementations of real-time AI-assisted fashion design applications.

One of the most popular program has been created through Project Muze made by a joint effort between Google and Zalando. Project Muze was a browser based web application where the full outfit for a displayed 3D avatar was generated by the AI system conditioned on multiple user inputs representing general interests such as favourite lifestyle, music, colour etc. This application presented many output limitations such as low variety, low user-output control, low quality as well as a high level of abstraction.

The solution proposed here attempts to overcome the mentioned weaknesses through an AI-driven web application

where users can have a high level of output control and where output quality is fostered by ad-hoc UI input guidelines.

III. Technical Implementation

This project is composed of three main pillars, first model understanding and selection, second model building through dataset creation and model training, and third web application technologies and model deployment.

A. Model Understanding

A generative adversarial network (GAN) is a deep neural network architecture that is made up of a generator and a discriminator network [9]. Those two networks interact and learn by trying to outwit each other during training.
On the one hand, the generator network uses existing data taken from a provided input distribution (or randomly generated vector of numbers, also known as latent space) to generate new data. In particular, the generator network G captures the input data distribution. For instance, when applied to computer vision tasks, the generator can generate new images or videos by using existing ones.
On the other hand, the discriminator network tries to differentiate between the real data (e.g. input image) and the data generated by the generator network. In particular, the discriminator network D estimates the probability that a sample came from the input data instead of G. Conditional generative adversarial networks [10], or more simply cGANs, represent a class of deep generative models where the generation process can be conditioned by feeding additional information to the generator and discriminator networks during training. In cGANs G and D are conditioned on some extra information y which is fed to the network as an additional input layer. In particular, the input noise $p_z(z)$ and y are combined in a joint hidden representation and the mini-max game is represented by Eq 2.

$$\begin{aligned} min_G\, max_D V(D,G) = E_x pdata(x)\,[logD(x|y)] \\ + E_z pz(z)\,[log(1 - D(G(z|y)))] \end{aligned} \tag{1}$$

TextureGAN[11] is the conditional generative model selected for this project. This model is able to generate realistic images from input sketches with overlaid textures with a relatively higher accuracy when using different type of textures. Fig. 1 shows the different results obtained by TextureGan on a single handbag sketch.

B. Model Building

1) Dataset Creation: In order to build a conditional generative adversarial network able to generate new fashion design images, the first step is to build a training dataset. The training dataset is available on the PyTorch implementation of TextureGan[11]. The original datasets used to extract the sketch and image pair are:

- Handbags dataset: Amazon Handbag images from [iGAN project]

Fig. 1. Results of handbags TextureGAN. On the far left side it is visible the "ground truth" image from which the sketch was synthesized. On the first result column, a texture patch is also sampled from the original shoe. There are showed three additional results with diverse textures.

- Clothes dataset: clothes images from [Deep Fashion Dataset]

The methods used to generate sketches and texture patches are explained in TextureGan[11]. First, a segmentation mask is created for handbags and clothes. Since the handbags dataset purely consists of product images with white background, the segmentation is done by setting white pixels as background pixels, while the segmentation mask for clothes is already provided in the dataset [12]. In particular for handbags, sketches are generated using the deep edge detection method used in pix2pix [14, 15] while for clothes, Canny edge detection is applied on the segmentation mask. Data generation techniques such as xDoG[16] and Scribbler [17] are implemented to increase sketches image variations and to add synthetic images. On the other hand, texture patches are generated by cropping a small area within the foreground of the ground truth images. Moreover, as explained in TextureGan[11] for fine tuning the generative model a leather-like texture dataset, composed of 6,300 leather textures, is created from 130 high resolution images queried and manually filtered from Google.

2) Training: TextureGAN[11] has a two-stage training scheme, first pre-training and then fine tuning. In the pre-training phase, input edge and textures are extracted from the ground truth. The adversarial network has been pre-trained with a combination of pixel, color, feature and adversarial losses. The parameters setup used follow the same provided on TextureGan[11]. The second stage is the texture fine-tuning used to imitate textures for which there are no ground truth examples. Compared to the previous stage, all losses are optimized at the same time but using different weights in order to compare the results with the entire input texture from which input texture patches are extracted. A first training iteration using a sample set of handbags images and a relatively small number of epochs (10) has been tested on a Jupyter notebook on a local machine. Finally, in order to save time, computing resources as well as energy, pre-trained representations of handbags and garments have been downloaded from [18] and preferred to use for the final deployment instead of training new models from scratch. Before deploying, the pre-trained model has been tested on a python notebook where few changes have been implemented. First the texture patch location instead of being placed each time the model is running at a random location of the sketch image, it has been modified so that the texture patch is always placed at the center of

the sketch image. This change is fundamental to guarantee consistent results with the user interface input guidelines provided in the web app.

C. Web Application Technologies and Model Deployment

The web application developed in this project works in two phases, first the acquisition of the user input as the sketch and texture images, and then the generation and presentation of the final output as the generation of a completed 2D fashion design image.

1) Web application technologies: In order to deploy a pre-trained model into a real-time web application, this project explored two major technologies: a web framework and a cloud environment.

Flask is the web application framework used to build the web application and Python Anywhere is the cloud hosting environment. Figure 2 summarises how the two technologies are integrated together to perform the fashion design image generation task.

Flask is a lightweight but very powerful server-side web framework and it is all in Python. Flask will automatically integrate with any Python work by passing all the machine learning and coding objects to the web. Flask is considered a micro web framework since it does not come with databases, form controls and any other components where pre-existing third-party libraries provide common functions. However, this lack is addressed by the fact that Flask supports extensions that can add application features as if they were supported in Flask.

Python Anywhere is an online integrated development environment and web-hosting platform. This online service allow to easily upload this project's file through an intuitive browser's interface. Web applications hosted by this service can be written in web server getaway interface (WSGA) based application framework such as Flask. This is one of the main reason why Flask and Python Anywhere are chosen for this project as they integrate well with each other.

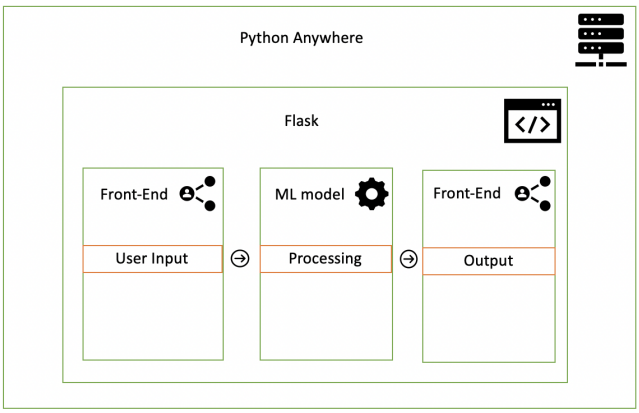

Fig. 2. Real-time, ML-driven web application hosted on Python Anywhere and built with Flask.

2) Model Deployment: In order to avoid installing all the required libraries on our local machine, a python virtual environment has been installed. The project structure is summarised in Figure 3.

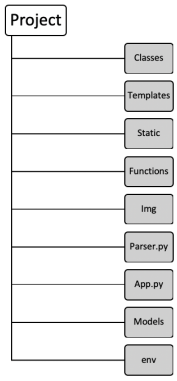

Fig. 3. Project structure.

The folders are explained as follow:

- Templates: Folder where all the HTML files are stored
- Static: For all the images used within the UI
- Functions and Classes: Web app functions and classes performing the image transformations and generations are stored here
- Env: This is the python virtual environment folder where the needed libraries are installed
- Img: This is the main folder containing other subfolders where the input images provided by the user are saved
- App.py: Represents the core file of this project. It contains the script to run the web application, including views and the main TextureGan function
- Models: This is the project folder containing the pre-trained models for the image generation of fashion design images of garments and handbags

App.py is the engine of this web application. This file contains the functions called through the user interface based on user actions. The logic of this file is organised and implemented according to the following structure:

- Sketch upload: A POST method is implemented to allow that image files (.jpg or .png) uploaded by users are saved into a temporary folder
- Texture upload: The same logic as the one explained for the sketch upload is applied for texture images while the file is stored into a different folder
- Image generation: This function is triggered by a UI button which, when clicked, it will call the TextureGan model conditioned on the two user inputs. The function will return the output image as an attachment

Finally, this project has been deployed on Python Anywhere where all the required libraries have been installed on their virtual environment. The final result of this project is a real time AI-assisted fashion design web application that can be accessed at: https://schedio.pythonanywhere.com/

IV. TESTING

The web application in question is able to create finished fashion design images of shirts and handbags. Figure 4 shows the three main screens of the final application which includes a home screen, handbags sketch upload, shirts sketch upload, texture upload and image generation.

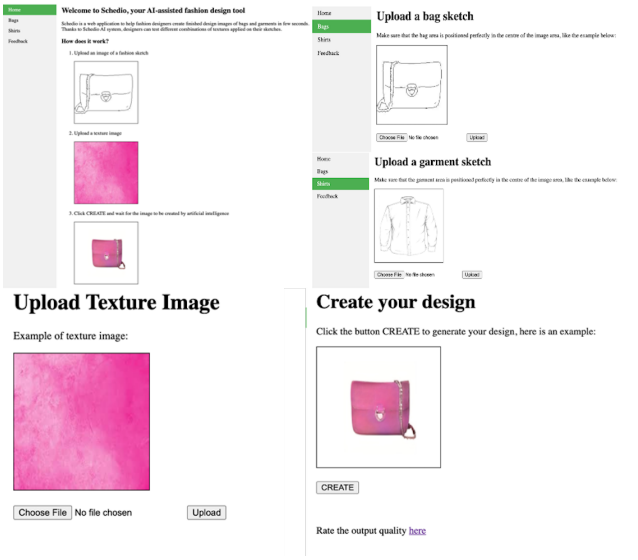

Fig. 4. Web app screens accessible at: https://schedio.pythonanywhere.com. From top left to bottom right: homepage, sketch upload page, texture upload page, design creation page.

The web application has been tested on MacBook Pro and iPhoneX, using a Safari browser. In this way, testing has been conducted on both a mobile and a desktop screen. The main functionalities as well as the design image generation have been tested using 6 sketches (3 shirts and 3 handbags) and 3 different textures (stripes, regular and irregular pattern), previously unseen during training and downloaded from Google Images. Examples of generated images based on sketches and textures never seen by the model during training are provided in Figure 5.

There are 5 different dimensions under which the web app has been tested.

- Portability. This is measuring how well the app is performing if accessed from different devices. This dimension has been tested mainly from two devices a desktop and a mobile phone. Overall, the level of portability is high since the functionality of image generation is working well both from desktop and mobile.
- Reality of the output. This dimension is assessing how close to a real fashion design image one of our output is. In order to test this dimension, multiple trials have been run using a test set of different combinations of sketches and textures. The different combinations have been compared to each other.
- Quality of the output. The quality of the output image has been assessed through a user survey linked to the web application. The survey is asking users to rate from 1 to 10 the quality of the output image.
- Speed. This metric is used to evaluate how fast is the image generation process which is calculated from the time the button to generate a new fashion design image is pressed until when the image is generated and provided as attachment to the end user.
- Easiness. This indicator is used to evaluate the user experience in the eyes of end users. The goal of this metric is to understand possible confusions caused by the frontend design. This is assessed through a free form feedback provided in the same user survey described above.

Where a user survey has not been used to evaluate the mentioned 5 dimensions, it has been used an empirical evaluation approach.

Fig. 5. Testing of the web application with shirts and handbags fashion design image generation using 3 different types of fashion sketches and texture images.

V. Analysis of the Results

Most of the dimensions mentioned in the previous section have been tested and analysed using an empirical approach.

- Reality of the output. The generated images look closer to real shirts or handbags design. Indeed, the uploaded sketch images are not strongly compromised during the generation process.
- Quality of the output. The empirical analysis has shown how results look better for irregular and regular texture patterns while stripes are not applied correctly for both shirts and handbags. Indeed, regular texture patterns are applied homogeneously giving a real look to the result. Moreover, results on short sleeves shirts, for irregular texture patterns, perform better than long ones. This is a limitation of the TextureGan model used which could be solved if the texture patch would be applied on the respective sleeves instead of only the main garment body or if a new TextureGan would be trained on a dataset of shirts only.
- Speed. In terms of image generation speed, given a stable internet connection, the current outputs are generated and provided as image attachment within less than two seconds (this is valid both for mobile and desktop).
- Portability. The web application is fully accessible both from mobile and desktop without any limitation.

Apart from the empirical approach explained above, user feedback has been collected with a user survey accessible through a link on the website. 30 Users provided their feedback where they could rate the output quality and write a free-form comment where the insights about the easiness of use of the web app have been analysed. The column chart showed in Figure 6 represents the output quality score which was asked to users on a scale from 1 to 10. A score of 7 represents our mode and median value, while the mean value is 6.67.

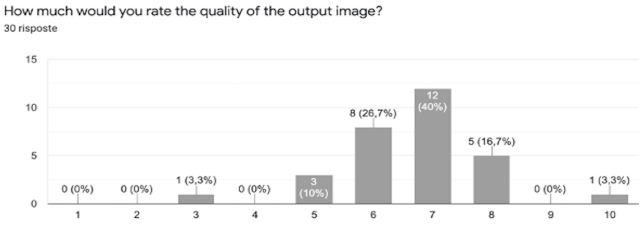

Fig. 6. Project survey results.

This is an indicator that can be interpreted as the output quality of being sufficiently satisfying but still subject to improvements. From the results obtained from the survey, it can be concluded that there are four possible areas of improvements.

- Increase image resolution: Image resolution has been identified as one of the major point of dissatisfaction. Indeed, providing a higher resolution output image (currently at 100 pixels/inch) would probably lead to better qualitative output results. This is something that can be done on the background. However, the trade off here is that a higher size and resolution output will cause the application to loose execution speed so the right point of balance between the two has to be applied.
- Output quality: The output quality depends on the type of texture that a user provides. In order to avoid low quality results, textures proven successful could be provided for selection from a drop-down menu implemented within the UI. Moreover, to avoid unrealistic results (e.g. on long sleeves shirts) a drag and drop UI functionality could be implemented so that users could drag a texture and drop it on the preferred garment area.
- CSS: The user interface has not been heavily elaborated but a more elaborated design could be implemented in future versions.
- UX: The user experience is also one of the element that can be better elaborated in future development of this web app. Indeed, for quicker creations, the whole image generation flow could all be merged into a single screen instead of multiple ones.

VI. Conclusion

This research has shown how TextureGAN could be embedded into a real-time web application (https://schedio.pythonanywhere.com/) and produce new fashion design images based on multiple user inputs. Despite a proven fast generation process, the output image, especially for irregular texture patterns, lacks of design quality. Because of this limitation further research, could be focused on finding a model that perform well also on non-regular texture images and on limiting textures to regular patterns only, selectable directly from the web app UI.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative adversarial nets.* In NIPS, 2014.

[2] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox and J. Clune. *Synthetizing the preferred inputs for neurons in neural networks via deep generator networks.* In NIPS, 2016.

[3] S. Zhu, S. Fidler, R. Urtasun, D. Lin, C. Change Loy. *Be your own Prada: fashion synthesis with structural coherence.* In ICCV, 2017.

[4] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, E. Shechtman. *Toward multimodal image-to-image translation.* In NIPS, 2017.

[5] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. *Generative adversarial text-to-image synthesis.* In ICMR, 2016.

[6] Y. R. Cui, Q. Liu, C. Y. Gao, and Z. Su. *FashionGAN: Display your fashion design using conditional generative adversarial nets.* In Computer Graphics Forum, 2018.

[7] G. Yildirim, C. Seward, U. Bergmann *Disentangling multiple conditional inputs in GANs* In KDD, 2018.

[8] O. Sbai, M. Elhoseiny, A. Bordes, Y. LeCun, C. Couprie *DesIGN: Design Inspiration from Generative Networks*.

[9] K. Ahirwar *Generative Adversarial Networks Projects: Build next-generation generative models using Tensorflow and Keras* Packt Publishing, January 2019.

[10] M. Mirza, S. Osindero *Conditional Generative Adversarial Nets* , arXiv:1411.1784v1, November 2014

[11] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, J. Hays *TextureGAN: Controlling Deep Image Synthesis with Texture Patches.* arXiv:1706.02823, 2017.

[12] C. Lassner, G. Pons-Moll, and P. V. Gehler. A generative model for people in clothing. In Proceedings of the IEEE International Conference on Computer Vision, 2017

[13] X.Liang,C.Xu,X.Shen,J.Yang,S.Liu,J.Tang,L.Lin,and S. Yan. Human parsing with contextualized convolutional neural network. In Proceedings of the IEEE International Conference on Computer Vision, pages 1386–1394, 2015.

[14] S. Xie and Z. Tu. Holistically-nested edge detection. In Proceedings of IEEE International Conference on Computer Vision, 2015.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image- to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004, 2016.

[16] H. Winnemo Ller, J. E. Kyprianidis, and S. C. Olsen. Xdog: an extended difference-of-gaussians compendium includ- ing advanced image stylization. Computers Graphics, 36(6):740–753, 2012.

[17] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scrib- bler: Controlling deep image synthesis with sketch and color. Computer Vision and Pattern Recognition, CVPR, 2017.

[18] Xian, Wenqi and Sangkloy, Patsorn and Agrawal, Varun and Raj, Amit and Lu, Jingwan and Fang, Chen and Yu, Fisher and Hays, James. Texturegan: Controlling deep image synthesis with texture patches. Retrieved from: https://github.com/janesjanes/Pytorch-TextureGAN

APPENDIX A

LITERATURE REVIEW

Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

School of Electronic Engineering

AI-assisted fashion design with deep generative adversarial networks

Literature Survey

Saverio Pulizzi
ID Number: 18210293

January 2020

MEng in Electronic and Computer Engineering

Supervised by Professor Kevin McGuinness

Declaration

I hereby declare that, except where otherwise indicated, this document is entirely my own work and has not been submitted in whole or in part to any other university.

Signed:  Date: 26/01/2020

IMPORTANT NOTE ON REFERENCES

One area that some students fall down on is the area of references. The guideline here is quite simple: either you did the work and wrote the text, or *someone else did*. For any element of your dissertation, even the tiniest element, which falls into the latter category, you must provide as complete a reference as possible, so that another researcher can easily access exactly the same source of information as you have. The desired form for the reference data is that used in IEEE conference papers and journals. Any direct quotes from another document **must** be placed between quotation marks and be followed by a reference.

There is additional information on the Loop site with respect to plagiarism and referencing; ignorance of this information is not a defence.

(Please delete this note when using this document as a template – it was inserted into a borderless text box for ease of deletion.)

AI-assisted fashion design with deep generative adversarial networks

Saverio Pulizzi

***Abstract*—Can fashion designers rely on a 2D web application to generate new design of fashion garments? In this research project, we will deploy a state-of-the-art generative model on a live web environment, where users will be able to generate a brand new fashion design based on user-selected shape and texture. The project will first provide an overview on generative adversarial networks and then it will compare three existent conditional generative adversarial networks, selecting the most feasible one, able to satisfy the objectives of our web app. As a conclusion, the project will deep dive into the technology used to build a web application as well as on some of the technical limitations of it.**

***Index Terms*—generative adversarial networks, web application, fashion design, AI system**

I. Introduction

NOWADAYS, bulky and sophisticated software represent the state-of-the-art technology used by fashion designers to create new design prototypes. A fast and reliable 2D solution to fashion design prototyping does not exist yet, while significant time and human capital investments are made by fashion companies towards the usage of existent garments modelling software. Traditionally, the design process relies on a pattern maker who is responsible for the selection of the texture (together with the material and the fabric) and a sample maker who works mainly on the shape of the product [8]. What if fashion designers could accelerate and augment their prototyping process with the aid of an artificial intelligence system? Imagine you could create hundreds of different combinations of colors and textures of your fashion sketch by simply uploading a picture of your draw. In few seconds, you would be able to create different styles for your sketch and select the ones that aesthetically match with your tastes. The goal of this paper is to develop a web application that can generate a fashion design image by automatically combining a selected texture and a picture of a fashion sketch. The output will be generated by leveraging existing machine learning models that have been proved successful in preserving structural coherence. Those models are able to preserve the shape of the uploaded fashion sketch while matching it with the selected texture conforming to the uploaded sketch outline. Deploying an existing machine learning technology into a real time web application is a very complex task, and the main goal of this project is to have our model running real-time while providing satisfying visual results to the final user. As a first step, the project will review some existing generative machine learning models. There are several machine learning models that generates new creatives from a given input and picking the one that can guarantee robustness of the visualized results will represent a key part of this project. Once selected the most suitable theoretical model for the application use case, training, testing and validating will take place accordingly to the procedures provided by its authors. At the end of this project, it will be covered the full model deployment phase into a web app with a user interface that will allow any user to upload an image of a sketch and to select a type of texture that will be processed by the machine learning system. Following the mentioned user inputs, the final output will be a complete textured 2D fashion design image.

The author would like to thank the project supervisor K. McGuinness who is with the school of Electronic Engineering, Dublin City University, Dublin, Ireland

II. Review and analysis of prior work

This section will explore different machine learning models inspired by the generative adversarial network proposed by Goodfellow et al. [1]. In particular, it will focus on those generative models applied to the fashion industry and trained based on conditions. Conditional generative adversarial networks are based on the principle that conditions incorporate additional information that influence the generation process. Conditions can be in the form of category labels [2], spatial configuration [3], encoded feature vector [4] or text [5]. This section will first start by giving an overview of GANs models and of their principal components. Afterwards, four different existing image generative models will be introduced, finally selecting the one that can prove to be the most suitable for the scope of this project's web application.

A. Generative Adversarial Networks

A generative adversarial network (GAN) is a deep neural network architecture that is made up of a generator and a discriminator network [9]. Those two networks interact and learn by trying to outwit each other during training.

On the one hand, the generator network uses existing data taken from a provided input distribution (or randomly generated vector of numbers, also known as latent space) to generate new data. In particular, the generator network G captures the input data distribution. For instance, when applied to computer vision tasks, the generator can generate new images or videos by using existing ones.

On the other hand, the discriminator network tries to differentiate between the real data (e.g. input image) and the data generated by the generator network. In particular, the discriminator network D estimates the probability that a sample came from the input data instead of G. During training, both networks provide feedback on the successful changes applied to their own processes (generation and discrimination) up to a point

at which the discriminator network is no longer able to identify the difference between a new generated data and the real one, reaching an optimal state also known as Nash Equilibrium.
Given an input noise and its related distribution $p_z(z)$, the generator network G is a differentiable function with parameters θ_g which maps $p_z(z)$ to the given data space x as $G(z;\theta_g)$ to learn and generate the distribution p_g. The discriminator network $D(x;\theta_d)$ outputs the probability $D(x)$ that the distribution came from the training data rather than p_g with the objective of maximizing the probability of assigning the correct label to training examples and samples from G. During training, G and D play a mini-max game with value function $V(G,D)$ as showed in Eq 1, where the parameters for G are adjusted in order to minimize $log(1D(G(z))$ and the ones for D to minimize $logD(x)$.

$$min_G\, max_D V(D,G) = E_x pdata(x)\,[logD(x) + E_z pz(z)\,[log(1 - D(G(z)))] \tag{1}$$

B. Conditional Generative Adversarial Networks

One of the major challenges of traditional GANs is that the generation process cannot be conditioned. Therefore, new generated data can be of any type and shape that are only determined by the underlying training data, excluding any influence from external sources. Conditional generative adversarial networks [10], or more simply cGANs, represent a class of deep generative models where the generation process can be conditioned by feeding additional information to the generator and discriminator networks during training. In cGANs G and D are conditioned on some extra information y which is fed to the network as an additional input layer. In particular, the input noise $p_z(z)$ and y are combined in a joint hidden representation and the mini-max game is represented by Eq 2.

$$min_G\, max_D V(D,G) = E_x pdata(x)\,[logD(x|y)] + E_z pz(z)\,[log(1 - D(G(z|y)))] \tag{2}$$

In the following subsections are presented three of the most recent cGANs techniques that can be applied to image generations.

1) BicycleGAN [4]: The vast majority of conditional image generation models have been focusing on single image output while BicycleGAN focuses on producing multiple realistic and diverse outputs. The major task performed by the BicycleGAN is multi-modal image-to-image translation. This represent one of the major strength of this model as most of the available CGANs focus on a single image to image approach.
The methodology used to produce both realistic and diverse results consists in modelling a distribution of potential outputs in the target domain (corresponding to multiple images), given an input image from one domain. An example of the output that this model is capable to achieve is showed in Figure 1.

From both a quantitative and qualitative perspective, BicycleGAN can guarantee a relatively good level of results with very diverse and realistic design images. Nevertheless,

Fig. 1. Output example of BicycleGAN in action; Credits:[4]

it does not represent the best fit for the use case of this project as there is the need for a model where image-to-image transformations can be applied with user controllable parameters and in particular where both shapes and textures can be decided by the user directly from the web interface.

2) StyleGAN [8]: The team of researchers who developed StyleGAN have been focusing on testing different architectures and losses with the goal of building a generative model that could generate both original and realistic fashion images deviating from the ones already appearing on the training set. In the StyleGAN model, a generator is trained to compute realistic images based on a mask input representing the shape of the fashion design image and a noise variable representing its style. One of the peculiarity of StyleGAN is its ability to sample different textures for the same shape by avoiding a deterministic mapping during the training of the generator and by introducing an additional $l1$ loss on the generator. An example of the output that this model is capable to achieve is showed in Figure 2.

Fig. 2. Output example of StyleGAN in action; Credits:[8]

While the results obtained from StyleGAN are very interesting from a qualitative perspective (measured on likeability and level of real appearance) of the generated fashion design image, the inability to condition this network on multiple conditions, such as shape and texture, represents a major limitation which makes this model not feasible for the purpose of our web application.

3) FashionGAN [6]: FashionGAN is a generative architecture which takes inspiration from BicycleGAN [4] but instead of producing multiple results from a single input, it takes two inputs (e.g. texture and shape) to generate just one output (process also called two-to-one mapping) among three image

domains. One of the major architectural difference between FashionGAN and BicycleGAN is related to its encoding strategy. Indeed, to ensure that different types of textures (even the ones not contained in the training dataset) are correctly applied to the shape, texture images are encoded in a latent vector with the goal of training the network with a fabric pattern image together with the ground truth and the contour image. Fig.3 shows how the input texture and image shape are combined together to generate a final complete image.

Fig. 3. Output example of FashionGAN in action with a list of latent vectors applied to generate diverse results; Credits:[6]

From a qualitative point of view, the output of FashionGAN results on average to be the highest in comparison to other similar models. In particular, the diversity and quality of the generated images has been measured with the Inception score across multiple dimensions such as: contour with cloth patch, contour with ground truth, fashion sketch with cloth patch and contour with stripe patch. In this project, the web application will generate a single output based on texture and shape of a fashion sketch which makes FashionGAN the most suitable model.

C. Model Deployment and web application technologies

In order to deploy a trained model into a real-time web application, this project will explore two major technologies: a web framework and a cloud environment.

1) Flask - web framework : Flask is a lightweight but very powerful server-side web framework and its all in Python. Flask will automatically integrates with any Python work by passing all the machine learning and coding objects to the web. Flask is considered a micro web framework since it does not come with databases, form controls and any other components where pre-existing third-party libraries provide common functions. However, this lack is addressed by the fact that Flask supports extensions that can add application features as if they were supported in Flask. The glue between Flask and HTML is JinJia2 which is one of the most used template engine for Python.

2) Python Anywhere - web-hosting platform: Python Anywhere is an online integrated development environment and web-hosting platform. This online service allow to easily upload this project's file through an intuitive browser's interface. Web applications hosted by this service can be written in web server getaway interface (WSGA) based application framework such as Flask. This is one of the main reason why Flask and Python Anywhere are chosen for this project as they integrate well with each other.

III. Review of prior work related to the project problem

The ultimate goal of this project it to build an ML-powered web app with a simple user interface where users can upload a shape and select a texture image directly from the web interface in order to generate a finished garment design. BicycleGAN and StyleGAN are not suited for this specific solution since the first produces multiple outputs based on a single input while the latter even though produces a single result, it can be conditioned on only one input. As a matter of fact, FashionGAN is the machine learning model that will be used to generate design images and it will be built following the same approach specified on the related paper. However, FashionGAN does not come with any limitation. Indeed, one of the major weakness of FashionGAN is that irregular fabric patterns are not mapped correctly on the garment shape and this cause the generation of very low quality outputs. In order to address this limitation, fabric patterns will be chosen directly from the user interface and the selection of patterns will be limited only to those textures on which the model has been trained on and proven to produce high quality outputs. In this way, the user experience will be preserved while avoiding the generation of low quality results. Training, validation and testing of the model is done on a Jupiter notebook running on a virtual machine and the datasets used will be the same used to train FashionGAN. The model file will be deployed on a web-hosting platform called Python Anywhere and it will be called from the interface of a web application built using a web framework called Flask. The mix of those technologies will result into a real-time, AI-driven web application where users will be able to generate new fashion design images based on their inputs as shown in Fig.4.

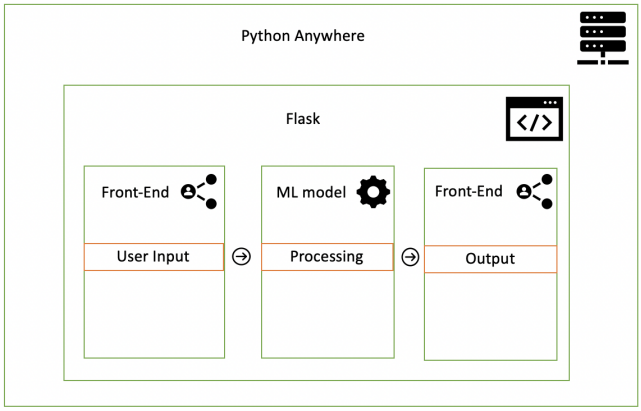

Fig. 4. Real-time, ML-driven web application hosted on Python Anywhere and built with Flask.

IV. Conclusion

There are many conditional generative adversarial networks that offer different results, however, only one model suits this

project use case. This research has shown how FashionGAN could be embedded into a real-time web application and produce new fashion design images based on multiple user inputs. Because of the limitations of the underlying generative model used, only few texture images will be available to select from the web interface and this represent one of the main limitation of this project. Further research, will be focused on finding a model that perform well also on non-regular texture images and to include more texture options as the current model will be trained on more and more patterns.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative adversarial nets.* In NIPS, 2014.

[2] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox and J. Clune. *Synthetizing the preferred inputs for neurons in neural networks via deep generator networks.* In NIPS, 2016.

[3] S. Zhu, S. Fidler, R. Urtasun, D. Lin, C. Change Loy. *Be your own Prada: fashion synthesis with structural coherence.* In ICCV, 2017.

[4] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, E. Shechtman. *Toward multimodal image-to-image translation.* In NIPS, 2017.

[5] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. *Generative adversarial text-to-image synthesis.* In ICMR, 2016.

[6] Y. R. Cui, Q. Liu, C. Y. Gao, and Z. Su. *FashionGAN: Display your fashion design using conditional generative adversarial nets.* In Computer Graphics Forum, 2018.

[7] G. Yildirim, C. Seward, U. Bergmann *Disentangling multiple conditional inputs in GANs* In KDD, 2018.

[8] O. Sbai, M. Elhoseiny, A. Bordes, Y. LeCun, C. Couprie *DesIGN: Design Inspiration from Generative Networks*.

[9] K. Ahirwar *Generative Adversarial Networks Projects: Build next-generation generative models using Tensorflow and Keras* Packt Publishing, January 2019.

[10] M. Mirza, S. Osindero *Conditional Generative Adversarial Nets* , arXiv:1411.1784v1, November 2014

APPENDIX B

PROJECT PLAN

Project Plan

Project Title: AI-assisted fashion design with deep generative adversarial networks
Student: Saverio Pulizzi 18210293
Supervisor: Kevin McGuinness

Research Question

Can a web application be used to generate new design of 2D fashion garments images?

Project Scope

- Conditional generative adversarial networks
- Dataset building
- Machine learning model training
- Machine learning model testing
- Machine learning model deployment
- Full stack web app development with python

Design Approach

- Review of the literature and current state-of-the-art conditional generative adversarial networks applied to the fashion industry
- Collection of fashion images and dataset building of texture and contour images
- Training and testing of a conditional generative adversarial network
- Build a web application
- Deploy the pre-trained model to make real time generation of fashion garments images

Detailed Timeline

Category	Milestone	Month	Task Description
Model building	Dataset building	March	Research activity aimed at finding a pre-built dataset needed to train the ML model.
Model building	Train, test, validate	April	Model building using python on Colab notebook. Training, testing and hyper-parameters tuning activity.
Web app	Build UI	May	Building the user interface components which allow a user to upload an image and select a texture.
Web app	Build UI backend handler events	June	Build the backend functionality to process the uploaded image together with the selected texture.
Deployment	Setup cloud environment and deploy model	July	Setup python anywhere cloud environment, deploying project folders and files.
Testing	Test web app and iterate	August/ September	Test the web app and iterate through feedback for bug fixes and product improvements.

Success Criteria

This project will be considered successful if the following criteria are met:

- Completion of the literature review
- Completion of the project presentation
- Completion of the project plan
- Completion of the web app
- Completion of the final portfolio submission

APPENDIX C

PROJECT RESEARCH LOG

Masters Project Research Log

Masters in Electronic and Computer Engineering 2019/2020

Student Name: Saverio Pulizzi
Student ID: 18210293
Project Title: AI-assisted fashion design with deep generative adversarial networks

Please read before making entries in this log

The purpose of this Project Research Log is to capture concise, focused summaries of research materials you read, as you progress through your project. The emphasis is to record (i) how the material you have read will determine or influence your project solution approach and (ii) your assessment of the key strengths and weaknesses of the solutions, methods, technologies, etc. proposed in the material you have read.

In the first stage of your project, the literature review, use the Log to capture this information for the key papers you have read (for example, the three most important papers of your 10 literature review references). As your project progresses into the design and implementation phases, you will need to continue to search the literature so you can review, revise and refine your initial thinking and the details of your approach to a project solution. Use this Research Log to capture your continued research reading and its influence on your project design and implementation.

Be selective about what you record in this log. Do not use it as an informal notebook while you are reading a new paper. Only make an entry after you have read a paper that you consider important to the development of your project solution. It is expected that, by the end of the project, you will have made **between 10 and 20 entries (20 maximum)**.

Your log will be shared with your supervisor for viewing throughout the project and you will submit the final version of the log for grading, at the end of the project implementation period. It will be assessed on the basis of how well you have used your analysis of the literature to inform your project design, implementation and the evaluation of your project results. The Research Log contributes **5%** to the overall project mark.

Note: All entries you make in this log must use the prescribed format. A new entry with the correct format is generated using the “Research Log” menu above (it may take a minute for this menu to appear). Do not make any other entries or change the format of the generated tables. You will maintain other notes as you progress through your project but they should not be recorded here. Do not exceed the maximum word counts indicated.

Statement of project problem / research question (maximum 200 words)

This statement should be periodically reviewed and updated, as necessary, as your project progresses and you gain further insight into the detailed project challenges, requirements and objectives as your project work moves from background reading, literature review, initial project design planning and detailed design and implementation. Initially, start by stating your current understanding of the project objectives. After each meeting with your supervisor, review and refine your project problem statement, as required.

Can a web application be used to generate new designs of 2D fashion images?

[Title]. Generative adversarial nets

[Full Bibliographic Reference] Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[Link to Online Resource]
https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Summary of paper (maximum 100 words)

Here is proposed a new framework for estimating generative models via an adversarial process, in which two models are simultaneously trained: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1/2 everywhere. Further experiments provided in the paper, demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)

This paper constitutes the base to understand generative adversarial networks which has been fundamental to be able to understand conditional GAN (the model used in my web application).

What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)

Strengths of this technology are that a model could be trained on a set of fashion images and be able to generate brand new fashion design images. However, weaknesses are that this model cannot be conditioned on other user input such as texture and shape of the design.

Log Entry Creation Date: Fri Feb 21 2020 12:16:24 GMT-0000 (GMT)

[Title] Toward multimodal image-to-image translation.

[Full Bibliographic Reference] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, E. Shechtman. Toward multimodal image-to-image translation. In NIPS, 2017.

[Link to Online Resource] https://arxiv.org/abs/1711.11586

Summary of paper (maximum 100 words)

In this paper, BicycleGAN is presented through the modelling of a distribution of possible outputs in a conditional generative setting. The ambiguity of the mapping is distilled in a low-dimensional latent vector, which can be randomly sampled at test time. A generator learns to map the given input, combined with this latent code, to the output. Mode collapse is prevented by explicitly encouraging the connection between output and the latent code to be invertible.

How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)

This paper presents a conditional generative adversarial network which is the type of model that we want to use to be able to generate new fashion design images on our web application conditioned on user input.

What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)

Strengths: The vast majority of conditional image generation models have been focusing on single image output while BicycleGAN focuses on producing multiple realistic and diverse outputs. The major task performed by the BicycleGAN is multi-modal image-to-image translation. This represents one of the major strengths of this model as most of the available CGANs focus on a single image to image approach.

Weaknesses: This technology does not represent the best fit for the use case of this project as there is the need for a model where image-to- image transformations can be applied with user controllable parameters and in particular where both shapes and textures can be decided by the user directly from the web interface.

Log Entry Creation Date: Wed Aug 26 2020 13:07:37 GMT+0100 (British Summer Time)

[Title] DesIGN:Design Inspiration from Generative Networks.

[Full Bibliographic Reference] O.Sbai,M.Elhoseiny,A.Bordes,Y.LeCun,C.Couprie DesIGN:Design Inspiration from Generative Networks.

[Link to Online Resource] https://arxiv.org/abs/1804.00921

Summary of paper (maximum 100 words)

In this paper, different image generation models are designed and investigated using different loss functions to increase the output variety and creativity in the generation process. There are investigated: (i) different Generative Adversarial Networks architectures that start from noise vectors to generate fashion items, (ii) novel loss functions that encourage novelty, inspired from Sharma-Mittal divergence, and (iii) a generation process following the key elements of fashion design (disentangling shape and texture components). As a result 61% of the created and tested images are thought to be created by a human designer.

How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)

This paper presents a different generative adversarial network architecture applied to the fashion design image generation.

What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)

Strengths: The team of researchers who developed StyleGAN have been focusing on testing different architectures and losses with the goal of building a generative model that could generate both original and realistic fashion images deviating from the ones already appearing on the training set. In the StyleGAN model, a generator is trained to compute realistic images based on a mask input representing the shape of the fashion design image and a noise variable representing its style. One of the peculiarities of StyleGAN is its ability to sample different textures for the same shape by avoiding a deterministic mapping during the training of the generator and by introducing an additional l1 loss on the generator.

Weaknesses: While the results obtained from StyleGAN are very interesting from a qualitative perspective (measured on likeability and level of real appearance) of the generated fashion design image, the inability to condition this network on multiple conditions, such as shape and texture, represents a major limitation which makes this model not feasible for the purpose of our web application.

Log Entry Creation Date: Wed Aug 26 2020 13:21:41 GMT+0100 (British Summer Time)

[Title] FashionGAN: Display your fashion design using conditional generative adversarial nets.
[Full Bibliographic Reference] Y. R. Cui, Q. Liu, C. Y. Gao, and Z. Su. FashionGAN: Display your fashion design using conditional generative adversarial nets. In Computer Graphics Forum, 2018.
[Link to Online Resource] https://www.researchgate.net/publication/328505097_FashionGAN_Display_your_fashion_design_using_Conditional_Generative_Adversarial_Nets
Summary of paper (maximum 100 words)
In this paper, a conditional generative adversarial network model able to generate fashion design image based on texture and shape conditions is presented. FashionGAN is a generative architecture which takes inspiration from BicycleGAN [4] but instead of producing multiple results from a single input, it takes two inputs (e.g. texture and shape) to generate just one output (process also called two-to-one mapping) among three image domains. One of the major architectural differences between FashionGAN and BicycleGAN is related to its encoding strategy. Indeed, to ensure that different types of textures (even the ones not contained in the training dataset) are correctly applied to the shape, texture images are encoded in a latent vector with the goal of training the network with a fabric pattern image together with the ground truth and the contour image.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
This paper presents a conditional generative adversarial network conditioned on fashion image shape and texture, which represents the type of model that we want to use to be able to generate new fashion design images on our web application conditioned on user input.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Strengths: From a qualitative point of view, the output of FashionGAN results on average to be the highest in comparison to other similar models. In particular, the diversity and quality of the generated images has been measured with the Inception score across multiple dimensions such as: contour with cloth patch, contour with ground truth, fashion sketch with cloth patch and contour with stripe patch. Weaknesses: The major weakness of FashionGAN is related to a low quality of the output when the model is conditioned with irregular texture patterns as well as stripes.
Log Entry Creation Date: Wed Aug 26 2020 13:38:19 GMT+0100 (British Summer Time)

[Title] TextureGAN: Controlling Deep Image Synthesis with Texture Patches.
[Full Bibliographic Reference] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, J. Hays TextureGAN: Controlling Deep Image Synthesis with Texture Patches. arXiv:1706.02823, 2017.
[Link to Online Resource] https://arxiv.org/abs/1706.02823
Summary of paper (maximum 100 words)
In this paper, a conditional generative adversarial network model called TextureGan is presented. The solution consists of a texture patch that can be placed on a sketch at arbitrary locations and scales to control the desired output texture. A local texture loss in addition to adversarial and content loss is developed to train the generative network. Tests show that this solution can produce more realistic results than adapting any existing methods directly.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
TextureGAN is the conditional generative model selected to deploy in our web application.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Strenghts: This model is able to generate realistic images from input sketches with overlaid textures with a relatively higher accuracy when using different types of textures also with irregular texture patterns. Weaknesses: The major limitation of this model is that performances in terms of output quality are very low when striped texture patterns are given as input to the network
Log Entry Creation Date: Wed Aug 26 2020 13:47:03 GMT+0100 (British Summer Time)

APPENDIX D

CODE

App.py is the core python script running the web app, the rest of the code is available at: https://github.com/spuliz/masterproject

App.py

```
import os
import glob
import shutil
# from creator import create
from flask import Flask, send_from_directory, flash, request, redirect, url_for,
render_template
from werkzeug.utils import secure_filename
from parser import parse_arguments

from functions.transformer import get_transforms
from functions.makeDataset import make_dataset
from functions.loadNetwork import load_network
from functions.getInput import get_input
from functions.normalize import normalize_lab, normalize_rgb, normalize_seg,
denormalize_lab, denormalize_rgb
from functions.vis import vis_image, vis_patch
from functions.getInputv import get_inputv
from classes.imageFolder import ImageFolder
from classes.textureGan import TextureGAN
from PIL import Image
import os
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np
import torch
from torch.utils.data import DataLoader


SKETCH_FOLDER = '/Users/spuliz/Desktop/schedio/img/val_skg/wendy'
VAL_SEG_FOLDER = '/Users/spuliz/Desktop/schedio/img/val_seg/wendy'
VAL_ERODED_FOLDER = '/Users/spuliz/Desktop/schedio/img/eroded_val_seg/wendy'
IMG_FOLDER = '/Users/spuliz/Desktop/schedio/img/val_img/wendy'
TEXTURE_FOLDER = '/Users/spuliz/Desktop/schedio/img/val_txt/wendy'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}




app = Flask(__name__, static_url_path='/Users/spuliz/Desktop/schedio/static')

# The absolute path of the directory containing images for users to upload
```

```
app.config['SKETCH_FOLDER'] = SKETCH_FOLDER
app.config['VAL_SEG_FOLDER'] = VAL_SEG_FOLDER
app.config['VAL_ERODED_FOLDER'] = VAL_ERODED_FOLDER
app.config['TEXTURE_FOLDER'] = TEXTURE_FOLDER
# The absolute path of the directory containing images for users to download
app.config["CLIENT_IMAGES"] = "/Users/spuliz/Desktop/schedio/img/output"

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/home', methods=['GET', 'POST'])
def home():
    return render_template('home.html')

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['SKETCH_FOLDER'], 'sketch.jpg'))
            for jpgfile in glob.iglob(os.path.join(SKETCH_FOLDER, "*.jpg")):
                shutil.copy(jpgfile, VAL_SEG_FOLDER)
                shutil.copy(jpgfile, VAL_ERODED_FOLDER)
                shutil.copy(jpgfile, IMG_FOLDER)
            # file.save(os.path.join(app.config['VAL_SEG_FOLDER'], 'sketch.jpg'))
            # file.save(os.path.join(app.config['VAL_ERODED_FOLDER'], 'sketch.jpg'))
            # return redirect(url_for('uploaded_file',filename=filename))
            return redirect(url_for('upload_texture'))
    return render_template('main.html')

@app.route('/garment', methods=['GET', 'POST'])
def upload_garment():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
```

```
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['SKETCH_FOLDER'], 'sketch.jpg'))
            for jpgfile in glob.iglob(os.path.join(SKETCH_FOLDER, "*.jpg")):
                shutil.copy(jpgfile, VAL_SEG_FOLDER)
                shutil.copy(jpgfile, VAL_ERODED_FOLDER)
                shutil.copy(jpgfile, IMG_FOLDER)
            # file.save(os.path.join(app.config['VAL_SEG_FOLDER'], 'sketch.jpg'))
            # file.save(os.path.join(app.config['VAL_ERODED_FOLDER'], 'sketch.jpg'))
            # return redirect(url_for('uploaded_file',filename=filename))
            return redirect(url_for('upload_cloth_texture'))
    return render_template('garment.html')



@app.route('/upload_texture', methods=['GET', 'POST'])
def upload_texture():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['TEXTURE_FOLDER'], 'texture.jpg'))
            # return redirect(url_for('uploaded_file',filename=filename))
            return redirect(url_for('profit'))
    return render_template('bag_texture.html')


@app.route('/upload_cloth_texture', methods=['GET', 'POST'])
def upload_cloth_texture():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
```

```
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['TEXTURE_FOLDER'], 'texture.jpg'))
            # return redirect(url_for('uploaded_file',filename=filename))
            return redirect(url_for('profit_cloth'))
    return render_template('cloth_texture.html')


@app.route('/profit', methods=['GET', 'POST'])
def profit():
    if request.method == 'POST':
        command = '--model texturegan --local_texture_size 50 --color_space lab'
        args = parse_arguments(command.split())
        args.batch_size = 1
        args.image_size =152
        args.resize_max = 256
        args.resize_min = 256
        args.data_path = '/Users/spuliz/Desktop/schedio/img' #change to your data path
        transform = get_transforms(args)
        val = make_dataset(args.data_path, 'val')
        valDset = ImageFolder('val', args.data_path, transform)
        val_display_size = 1
        valLoader = DataLoader(dataset=valDset, batch_size=val_display_size,
shuffle=False)
        # pre-trained model for handbags
        model_location =
'/Users/spuliz/Desktop/schedio/textureD_final_allloss_handbag_3300.pth' #change to
your location
        netG = TextureGAN(5, 3, 32)
        load_network(netG, model_location)
        netG.eval()
        data = valLoader.__iter__().__next__()
        color_space = 'lab'
        img, skg, seg, eroded_seg, txt = data
        img = normalize_lab(img)
        skg = normalize_lab(skg)
        txt = normalize_lab(txt)
        seg = normalize_seg(seg)
        eroded_seg = normalize_seg(eroded_seg)
        inp,texture_loc = get_input(data,-1,-1,30,1)
```

```
        seg = seg!=0
        model = netG
        device = torch.device("cpu")
        inpv = get_inputv(inp.to(device))
        output = model(inpv.to(device))
        out_img = vis_image(denormalize_lab(output.data.double().cpu()), color_space)
        plt.figure()
        plt.imshow(np.transpose(out_img[0],(1, 2, 0)))
        plt.axis('off')
        plt.savefig('/Users/spuliz/Desktop/schedio/img/output/output.png', dpi=1000)
        return send_from_directory(app.config["CLIENT_IMAGES"], filename='output.png',
as_attachment=True)
        return redirect(url_for('http://127.0.0.1:5000/profit/output.png'))
    return render_template('profit_bag.html')



@app.route('/profit_cloth', methods=['GET', 'POST'])
def profit_cloth():
    if request.method == 'POST':
        command = '--model texturegan --local_texture_size 50 --color_space lab'
        args = parse_arguments(command.split())
        args.batch_size = 1
        args.image_size =152
        args.resize_max = 256
        args.resize_min = 256
        args.data_path = '/Users/spuliz/Desktop/schedio/img' #change to your data path
        transform = get_transforms(args)
        val = make_dataset(args.data_path, 'val')
        valDset = ImageFolder('val', args.data_path, transform)
        val_display_size = 1
        valLoader = DataLoader(dataset=valDset, batch_size=val_display_size,
shuffle=False)
        # pre-trained model for handbags
        model_location = '/Users/spuliz/Desktop/schedio/final_cloth_finetune.pth'
#change to your location
        netG = TextureGAN(5, 3, 32)
        load_network(netG, model_location)
        netG.eval()
        data = valLoader.__iter__().__next__()
        color_space = 'lab'
        img, skg, seg, eroded_seg, txt = data
        img = normalize_lab(img)
        skg = normalize_lab(skg)
        txt = normalize_lab(txt)
        seg = normalize_seg(seg)
        eroded_seg = normalize_seg(eroded_seg)
        inp,texture_loc = get_input(data,-1,-1,30,1)
```

```
        seg = seg!=0
        model = netG
        device = torch.device("cpu")
        inpv = get_inputv(inp.to(device))
        output = model(inpv.to(device))
        out_img = vis_image(denormalize_lab(output.data.double().cpu()), color_space)
        plt.figure()
        plt.imshow(np.transpose(out_img[0],(1, 2, 0)))
        plt.axis('off')
        plt.savefig('/Users/spuliz/Desktop/schedio/img/output/output.jpg')
        return send_from_directory(app.config["CLIENT_IMAGES"], filename='output.jpg',
as_attachment=True)
        return redirect(url_for('http://127.0.0.1:5000/profit/output.jpg'))
    return render_template('profit_cloth.html')
```