Clothing Texture Synthesis using CycleGAN

MIGUEL FERRER AVILA (MFERRERA)
JI HOON "ANDY" KIM (JKIM4223)

Github Link:
https://github.com/JihoonAndyKim/CS230-Final-Project

1 MOTIVATION AND PROBLEM STATEMENT

In recent years, the availability of large amounts of data, a surge in computational power thanks to scalable and parallel systems, and more efficient algorithms have driven the explosive increase of machine learning applications in a variety of fields. Interestingly, one of the areas impacted is the fashion industry, with applications such as attribute and category matching ([Z. Liu 2016]) and deep learning recommendation systems ([S. Zhang 2018]).

Simultaneously, the advent of generative adversarial networks has allowed style synthesis and generation to be easily carried out by machine learning models. A subset of this algorithm is the CycleGAN architecture proposed by Zhu et al. ([J. Zhu 2017]), which performs unpaired image-to-image translation. In the present work, we propose the following question: Can we use the CycleGAN architecture to generate novel and new clothing designs? In particular, can we take a style domain, such as leather clothing, and translate it into another domain, such as denim clothing?

2 LITERATURE AND BACKGROUND

Before delving into the details of the network architecture and our model, we give some insight into the background of our problem.

Date et al. proposed a method to perform paired image clothing synthesis [P. Date 2017]. They trained a convolutional neural network using VGG-19 weights, along with content extraction from labels to generate clothing. However, we note that this training regime requires labeled paired images, which is not applicable to our problem of translating images between the domains of leather and denim clothing.

Instead of using CNN, S. Zhu et al. [S. Zhu 2017] had a slighly different approach by using a two-stage GAN framework to sythesize clothing styles from descriptions. The framework takes in descriptions, renders a segmentation map in the first stage and then uses this mapping with a DCGAN to produce an image with the parameters described. However, while this uses a generative model to create a new design for fashion, it still requires paired training results as well as performing a different style of problem (text-to-image generation rather than image-to-image translation).

Zhu et al. ([J. Zhu 2017]) proposed a method to perform unpaired image to image translation (translating from domain A to domain B) using a new architecture that performs training without the use of paired training data. This is enforced by a loss formulation that includes cyclic loss, i.e. an image from domain A translated into domain B and back to domain A should be the identical to the starting image. To further support this formulation as applicable to our task, Chu. et al [C. Chu 2017] used CycleGANs in their work to reconstruct aerial images from basic maps and vice versa by performing an image-to-image translation between the two domains. For this reason, we adopt the CycleGAN architecture as the most qualified model to perform our unpaired image-to-image translation between leather and denim clothing.

3 DATASET

While the ImageNet database presents a wealth of images for deep learning, a small subset of these images are actually fashion related and an even smaller subset are usable for domain transfer (i.e. well-lit images with an article of clothing in the foreground), making it a poor choice for this domain transfer problem. Instead, we use the DeepFashion database to procure images for training ([Z. Liu 2016]).

The pre-curated database contains 800,000 images of articles of clothing segregated by fabric, texture, shape and style. Each image is approximately 300×300 pixels with three channels for color. What makes this dataset much more viable for our problem is that the majority of the articles of clothing appear clearly in the foreground, while a white background helps to reduce the amount of noise. For our problem, we specify two datasets corresponding to the two domains between which we wish to translate: leather clothing and denim clothing.

3.1 Domain A: Leather Clothing

The leather dataset contains 646 images of leather clothing, with 548 used for training and 98 for testing. 75% of the dataset in both train and test sets is composed of upper-body articles of clothing, and the remaining is lower-body clothing. In addition, approximately 70% of our dataset are images of clothing without a person present in the image, while the rest are images of models wearing leather clothing. We discuss the impact of this imbalance in the results section. To combat the imbalance, we added additional samples of images with models at a later time to produce better training results (around 50 images).

3.2 Domain B: Denim Clothing

The denim dataset includes 711 images of denim clothing with 622 for training and 91 for testing. Approximately 75% of the dataset in both train and test sets is composed of lower-body articles of clothing with the remaining for upper-body clothing. Nearly 90% of our dataset are images of clothing without a person present in the image. Similarly, we added additional samples of images with models to this dataset as well (also around 50 images).

3.3 Data Processing

As per the training regime for CycleGAN proposed by Zhu et al ([J. Zhu 2017])), we resize all images to a square 286×286 image (since larger images become computationally expensive), perform random cropping to a 256×256 image, and randomly flip the image horizontally as well as rotate the images. This is to add stochasticity to our model, as well as to augment our dataset so that we do not overfit on the input images. In addition, we center and normalize the pixel values with a mean of 0.5 and a standard deviation of 0.5 so that we start with values around 0.5 rather than ranging from 0 to 256. Furthermore, our architecture is highly dependant on the input data format, as we explain in the next section.

4 METHODS

The architecture and details of our model follows the CycleGAN implementation as described by Zhu et al. This is outlined in Figure 1. We train over datasets A and B with two generators, G and F, and two discriminators, D_A and D_B. D_B helps G translate images from domain A to be indistinguishable to those in domain B, while D_A helps F translate images from domain B to be indistinguishable to those in domain A. Since we are experimenting with GANs, it is difficult to describe a baseline model; thus, we provide an different architectures we are using for training. First we describe the losses we use to train our model.

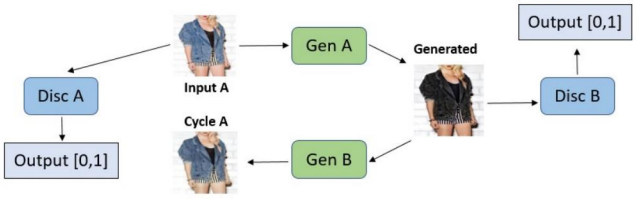

Fig. 1. Architecture of the CycleGAN model. Here generators G and F are Gen A and Gen B respectively and discriminators D_A and D_B are Disc A and Disc B respectively.

4.1 Loss Formulation

We employ a loss metric composed of the sum of the adversarial loss from each pair of generator and discriminator with the cyclic loss:

$$\mathcal{L}_{Total} = \mathcal{L}_{GAN}(G, D_B, A, B) + \mathcal{L}_{GAN}(F, D_A, A, B) + \lambda \mathcal{L}_{cyclic}$$

The $\mathcal{L}_{GAN}$ is defined as follows

$$\mathcal{L}_{GAN}(G, D_B, B, A) = \mathbb{E}[\log(D_B(b))] + \mathbb{E}[\log(1 - D_B(G(a)))] + \mathbb{E}[\log(D_B(G(a)))]$$

where the first two terms are related to discriminator loss (it is the binary cross entropy loss to distinguish the fake images from the real images), and the last term is related to the generator loss (we want to trick the discriminator to mis-identify an image, or produce a "0").

The cyclic loss $\mathcal{L}_{cyclic}$ is defined as

$$\mathcal{L}_{cyclic} = \mathbb{E}||x - F(G(x))||_1 + \mathbb{E}||y - G(F(y))||_1$$

where we take an image and feed it to both generators, hoping to recover the same image. Ultimately, our complete objective function to recover the optimal image generators is

$$(G^*, F^*) = \arg\min_{G,F} \max_{D_A, D_B} \mathcal{L}_{Total}$$

This is solved with backpropagation through the GANs and with ADAM optimizers.

4.2 Network Architecture

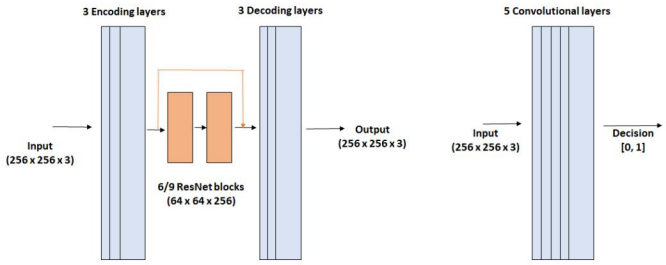

Fig. 2. **Left:** Architecture of the Generator. **Right:** Architecture of the Discriminator

The general network architecture we use is the same as that cited by Zhu et al; however, we made several changes to the generator. The architecture for both generator and discriminator are seen in Figure 2. Please reference the code to see the exact hyperparameters used.

4.2.1 Generator. Our generator takes in an image that is cropped to 256×256 with 3 channels as per the data processing step, and feeds it sequentially to the following layers.

Encoding. Three convolutional layers with ReLU activation and batch normalization. The layers bump up our images from 3 channels to 256 channels. The hyperparameters (kernel size, stride and padding) chosen for this model follow from the architecture described by Zhu et al.

Translation. This block of our generator is composed of ResNet blocks with skip connections between the i and $i + 2$ layers with Batch-norm and ReLU activation. We preserve the dimensions of each input and output of the blocks. This is where we differentiate our network architectures, as we take on multiple models with ResNet-6 and ResNet-9 (Each number corresponding to the number of ResNet blocks used in the model).

Decoding. Two deconvolutional layers with ReLU activation, and a final deconvolutional layer with tanh activation boosting back to the original dimensions in addition to Batch-norm on all of the inputs to the layers. This mirrors the encoding step.

Once the image is fed through the generator, the output should be an image in the new domain.

4.2.2 Discriminator. Our discriminator is composed of five convolutional layers (different from the PatchGAN proposed by Zhu et al) with four layers with Leaky ReLU activation boosting from 3 channels to 512 channels with a final convolutional layer to take all 512 channels and produce a one dimensional output. The image is fed through these five layers and to obtain a binary output that signals whether the image is fake or not. This was inspired by a blog post by Bansal ([H. Bansal 2017]). We performed preliminary experiments by varying the number of layers (adding another fully connected network and removing one of the convolutional layers in between) as well as different unit sizes for each layer, but noticed no appreciable difference between the qualitative output as well as between the losses of the training. Therefore, for this project, we adhere to the model proposed by Bansal.

4.3 Training Regime

For the training, we use the hyperparameters used by Zhu et al. as a starting point. Most importantly, for our loss formulation, we chose a $\lambda = 10$ (Since we want to penalize cyclic loss more) so that we weight the cycle loss much more than the adversarial loss. The optimizers we used are ADAM optimizers for backprop ($\beta_1 = 0.5$, $\beta_2 = 0.999$), a learning rate of 0.002, a leakyReLU coefficient of 0.2, a mini-batch size of 10 and 250 epochs for training. For the backpropagation, we use gradient clipping to ensure that our training regime stays on track.

In addition, we modify the loss formulation for training in that we perform least squares loss for $\mathcal{L}_{GAN}$ instead of the log likelihood since this provides stable training (faster training at first, but slower training at later epochs since our $D_B(G(Z))$ value will be close to 0 and we want a non-saturating cost).

After training several different models with randomized hyperparameters and using the full objective loss function as our metric to evaluate our different models, we note that still those we had picked initially performed with the least loss (Objective loss of 2.86 compared to 3.5 seen with other architectures). However, we note the best training results were those trained with more epochs.

5 RESULTS AND ANALYSIS

Fig. 3. Image-to-Image Translation from leather to denim with class imbalance

For this section, we compare two of the models that we tested, ResNet-6 and ResNet-9 (described in the Method section) and show the results from one of the models we chose between the two.

5.1 Qualitative Results

Initially, we started with two domains: handbags and dresses. However, training showed unpromising results, as the generated images simply appeared as color-shifted versions of the original image. This is likely due to the high variance within the domains themselves (many of the handbags look different from each other), and solving it would require much larger training sets. Therefore, we narrowed the focus to a translation between leather clothing and denim clothing. Generated images are shown from this task in Figure 4 after training the model given the regime described in the previous section.

Furthermore, we noticed that the translation from leather to denim resulted in a blurry and poor result as shown in Figure 3. This was due to dataset imbalance, as we had many more examples of humans in the domain for leather clothing than in the domain of denim clothing. Additionally, since facial features are rather complex, we require many training examples to produce a robust result. In consequence, we spent some time adding images to the dataset that included more human figures as a reference (Approximately 50 images for each domain) to reduce the overfitting on the non-model images. Qualitatively, we see a high-fidelity image with little loss on the facial structure of the original image as shown in the leftmost column of Figure 4.

Between the ResNet-6 and ResNet-9 architectures, there were no noticeable qualitative difference even after running the models for over 200 epochs during training. Therefore, we defer the judgment between the models with the quantitative results.

5.2 Quantitative Results

Since we do not have a ground truth for many of our outputs, it is difficult to quantify what a "good" output is. Thus, instead

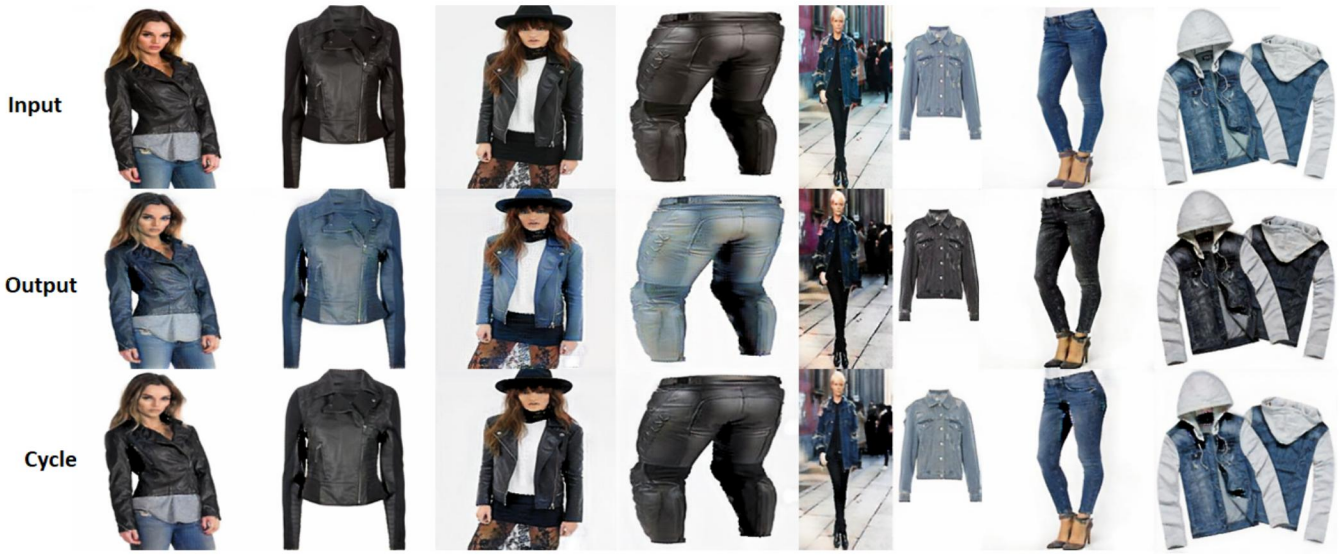

Fig. 4. Image-to-Image Translation output on our validation set. Top row shows original image, second row show generated image and third row shows the recovered image

of using typical metrics such as MSE loss, we evaluate our model based on the metric used to optimize the algorithm - the total objective loss.

Comparison between the ResNet-6 and ResNet-9 architectures. We run two different models: ResNet-6 and ResNet-9, which produced the following objective function plots, as described in the loss formulation section.

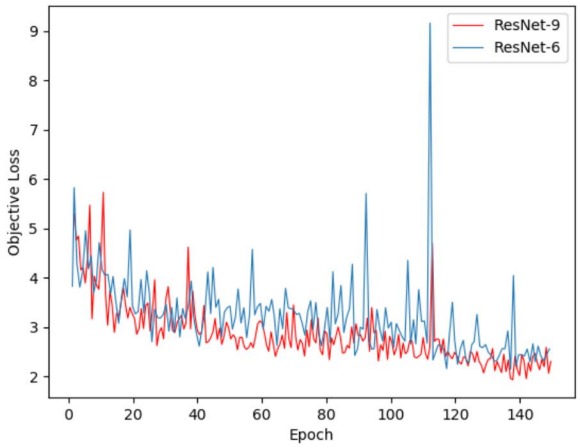

Fig. 5. The epoch number is plotted on the x axis and the objective function is plotted on the y-axis.

While the general trend of the full objective function loss is correctly decreasing, the plots shown in Figure 5 tell us there is room to further decrease the loss. This can be done with additional training over more epochs, or increasing the training set size (Split with less examples in the test set and include more for training). Another important point to note is that the ResNet-9 architecture ($\mathcal{L}_{Total,average}$: 2.86) performs better in general than the ResNet-6 ($\mathcal{L}_{Total,average}$: 3.22) architecture. This shows that a more complex model performs better on our training set. Hence, we move towards a more Panda approach in that we curate the model with a ResNet-9 architecture moving forward—since training CycleGAN is expensive and performing additional model search for our models is a computationally prohibitive task.

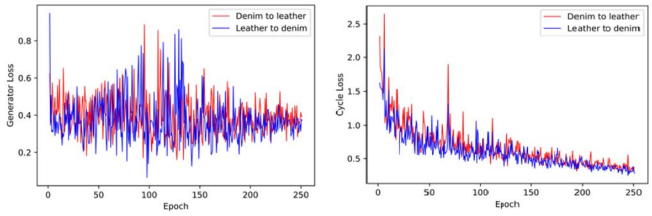

Fig. 6. **Left** - Generator Loss. **Right** - Cyclic Loss. The epoch number is plotted on the x axis and the loss is plotted on the y axis.

Cyclic, Generator and Discriminator Loss. In this section, we pull apart the adversarial loss to formulate both generator and discriminator loss. Figure 6 shows the cycle and generator losses respectively. Since we heavily weight the cyclic loss over the total objective by setting our loss function hyperparameter $\lambda = 10$, we see better minimization of the cyclic loss over epochs than that of the generator. After 250 epochs, we can see that the cyclic loss converges close to 0, which is to be expected. Comparing the generator loss in Figure 6 to the discriminator loss in Figure 7, we note that the generator loss does not decrease much. This is unsurprising, as the minimax game performed with the objective loss function comes to saturation eventually, when we create a discriminator that is not easily fooled by the generator.

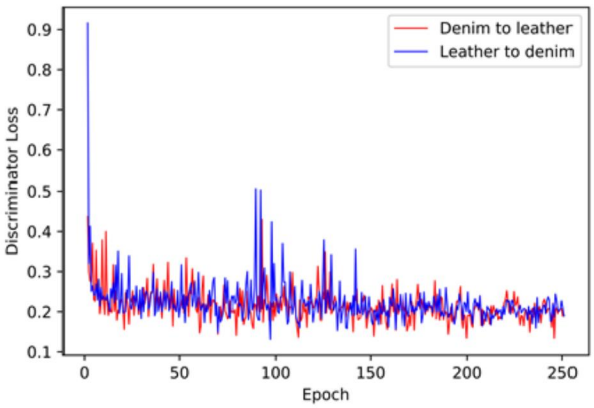

Fig. 7. Discriminator loss plot. The epoch number is plotted on the x axis and the loss is plotted on the y-axis.

A more interesting feature in the plot shown in Figure 6 is the bulge in loss between 100 and 150 epochs, which is also noticeable in the discriminator loss in Figure 7. This is because around 100 epochs, either the generator or discriminator was significantly stronger than the other at performing its assigned task of minimizing its adversarial loss, driving the loss of the other network high. However, after training successfully for 100 more epochs, we observe a sort of convergence between the discriminator (with loss hovering around 0.2) and the generator (with loss around 0.4). We also note this artifact of instability in Figure 5 around 110 epochs.

Since we see that all the losses of our minimax game converge, additional training beyond the 250 epochs performed would likely not improve the results. Therefore, we would need to spend more time on curating additional data for training or trying out different model architectures.

6 CONCLUSIONS AND FURTHER WORK

As explained in the Results section, we show that our loss curves for discriminator, generator and cyclic loss converge to appropriate values expected with a minimax objective function, hinting that additional training would likely not offer much difference. As seen in Figure 4, we generate images that illustrate, qualitatively, that our model works well in performing the task we aimed to solve: translating images between leather and denim clothing. We also demonstrated that, for our model, the ResNet-9 architecture performed better than the ResNet-6 architecture as well as using the hyperparameters outlined by Zhu et al. [J. Zhu 2017].

We adapted what we learned from the milestone and have several goals moving forward with the project:

(1) **Inception Scoring:** Instead of using the full objective loss, we would like to try using inception scoring as a metric for our CycleGAN, which would give us a better idea of the image diversity and the quality of the images (several of our images demonstrate blurring due to artifacts from training).
(2) **More Models:** Try more models other than ResNet-6 or ResNet-9 such as ResNet-50, DenseNet or UNet. Zhu et al. ([J. Zhu 2017]) mentions UNet works well.
(3) **Change discriminator:** We have not yet experimented modifying the discriminator, except for smaller side experiments by changing the number of layers and sizes. Perhaps changing the formulation will allow us generate higher fidelity images by incorporating the inception scoring metric above and/or using some of dense net architectures we reviewed in class.
(4) **Increasing training size further:** Although much time was spent curating the data and retrieving sample images to balance our training set, we showed that after 250 epochs, training was saturated and the loss curves were stagnant. Therefore, adding even more training data would aid our model achieving more quality results without overfitting.

7 CONTRIBUTIONS

Both team members worked on performing the background literature analysis, as well as interpreting the quantitative and qualitative results presented here. In addition, both team members worked on the project report, poster and presentation equally.

Miguel worked on coding up the CycleGAN implementation in PyTorch, as well as setting up and performing multi-GPU support with our codebase for fast training. He also worked on training the images on the compute node to produce the results in the paper.

Andy worked on data fetching and processing including scraping, cleaning and preparing the images from DeepFashion to be fed into the model. He also worked on training the model by curating the data and adding data augmentation in the codebase as well as performing additional coding and debugging work.

REFERENCES

M. Sandler C. Chu, A. Zhmoginov. 2017. CycleGAN, a Master of Steganography. *31st Conference on Neural Information Processing Systems (NIPS 2017)* (December 2017).

A. Rathore H. Bansal. 2017. https://hardikbansal.github.io/CycleGANBlog/. (2017).

P. Isola A. Efros J. Zhu, T. Park. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *IEEE* (2017).

T. Oates P. Date, A. Ganesan. 2017. Fashioning with Networks: Neural Style Transfer to Design Clothes. *ML4Fashion* (2017).

A. Sun Y. Tay S. Zhang, L. Yao. 2018. Deep Learning based Recommender System: A Survey and New Perspectives. *Comput. Surveys* 1, 1 (2018), 1–35.

R. Urtasun D. Lin C. C. Loy S. Zhu, S. Fidler. 2017. Be Your Own Prada: Fashion Synthesis with Structural Coherence. *International Conference on Computer Vision (ICCV)* (2017).

P. Luo X. Wang X. Tang Z. Liu, S. Yan. 2016. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. *European Conference on Computer Vision (ECCV)* (October 2016).