Sketch2Fashion: Generating clothing visualization from sketches

Manya Bansal
Stanford University
manya227@stanford.edu

David Wang
Stanford University
davidw23@stanford.edu

Vy Thai
Stanford University
vythai@stanford.edu

Abstract

The field of unsupervised image-to-image translation in computer vision has undergone several developments giving rise to models that produce high-quality images while overcoming the one-to-one mapping used in earlier models. By leveraging the ability of these models, we undertake a project that aims to simplify the process of fashion design while preserving the creativity that is critical to the process by transforming sketches of fashion designs to final outfits, complete with textures and patterns. Our project experiments with three edge detection algorithms and tests out three models with different architectures. We perform qualitative as well as quantitative analysis through a human perceptual study to note possible advantages and disadvantages of the three models as they relate to the goals of our project.

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

1 Introduction

Drawing sketches is the first part of any fashion design process. Our project transforms the sketch to a realistic, colored image of clothing that propels the process of designing clothes to its last stage: an image of a wearable piece of clothing that captures the subtleties of patterns, fabrics and textures. Through this project, we aim to facilitate creativity and provide ease and speed in the production of fashion designs. The input is a rough sketch of a piece of clothing, while the output is a realistic, colored image translated from the input sketch. We try multiple models (CycleGAN, CGAN, MUNIT) and edge detection algorithms (HED, Canny, CycleGAN) to determine which model is best suited to accomplish the goals of our project.

2 Dataset and Input Pipeline

We use an open-source fashion clothes dataset contributed by Leonidas Lefakis, Alan Akbik, and Roland Vollgraf [2]. The dataset consists of 8,792 images of dresses. Since the dataset doesn't contain the sketches for each dress, we generate "sketches" for each photo. We use three different methods to generate the sketches: HED, Canny, and the sketch-output of CycleGAN. We then split it into training (7033) and test (1759) sets [1]. Last but not least, we normalize the training examples and apply data augmentation techniques such as random flipping and cropping to increase the diversity of inputs and reduce overfitting. Since the dresses are placed on a white background, they requires a robust edge detection to generate the edges for light-colored dresses. The following is the summary of each algorithm's implementation and performance.

2.1 Holistically-Nested Edge Detection [5]

We run the HED scripts provided in CGAN repository to extract coarse edges from real clothes images. While the edges are generally good, they eliminate the details of the dresses and do not represent the general design sketch that the model will see in a human-drawn fashion design.

2.2 Canny Edge Detection [6]

We set $\sigma = 1.7$ for dark-colored dresses and decreased σ to ranges $\sigma \in [1, 0.6, 0.4]$ accordingly for bright to extremely-bright colored dresses. This is critical to avoid missing edges on white or light colored dresses. Although Canny generates more detailed sketches than HED, the edges still do not capture many important details in the dresses.

2.3 Sketch outputs of CycleGAN

Initially we only used edge detection algorithms, meaning our inputs were sparse, grayscale "sketches". However, while training CycleGAN on Canny edge input, we found that the outputted sketches generated by CycleGAN are closer to human-drawn fashion designs, making them more suitable for training than the ones generated by HED or Canny. To our surprise, the reconstructed dress images are extremely close to the ground truth. However, when we take a closer look into the sketches that it generates, it actually "cheats" by including additional color information (Figure 1). In order to use these sketches as inputs, we do additional pre-processing to convert them to grayscale and remove all the noise outside of the dresses. As expected, the FID scores of models that use these sketches as input as opposed to HED or Canny are superior, Refer to section 4.2 for a more detailed discussion of the FID scores.

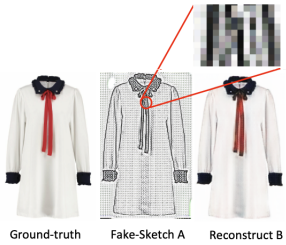

Figure 1: CycleGAN "cheats" by using colored pixels in the fake-sketch to inform the reconstruction process. This is why the reconstructed image almost perfectly replicates ground truth.

Thus, we decided to use CycleGAN-generated sketches as our inputs in order to compare different models in the following sections.

Figure 2: Different edge extraction performances after training with CGAN for 75 epochs

[1] We ran FID on them to make sure training set and test sets have a highly similar distribution

3 Architecture

3.1 MUNIT [4]

In order to generate various styles, we implement the Multimodal Unsupervised Image-to-Image Translation (MUNIT) Model. The model consists of two auto-encoders that are trained with adversarial objectives. The loss function consists of an image reconstruction loss including considerations for style (s) and content (c), a latent reconstruction loss and an adversarial loss to finally combine with weights and calculate a total loss. The image reconstruction loss is given by $\mathcal{L}^{x1}_{recon} = \mathbb{E}_{x_1 \sim p(x_1)}[||G_1(E^c_1(x_1), (E^s_1(x_1)) - x_1||_1]$. The latent reconstruction loss for content (there is as similiar loss for style) if given by $\mathcal{L}^{c_1}_{recon} = \mathbb{E}_{c_1 \sim p(c_1), s_2 \sim q(s_2)}[E^c_2(G_2(c_1, s_2)) - c_1||_1]$.The adversarial loss is given by $\mathcal{L}^{x2}_{GAN} = \mathbb{E}_{c_1 \sim p(c_1), s_2 \sim q(s_2)}[\log(1 - D_2(G_2(c_1, s_2)))] + \mathbb{E}_{x_1 \sim p(x_2)}[\log D_2(x_2)]$. See References [4] for more information.

We chose number of iterations= 200,000, batch size= 1, weight decay= 0.0001, β_1= 0.5, β_2= 0.999 , kaiming weight initialization, initial learning rate= 0.0001 , decay every 150000 iterations by 0.5 each time, adversarial loss weight= 1, image reconstruction loss weight= 10, style reconstruction loss weight= 1 and content reconstruction loss weight= 1. The model was trained for a total of 25 epochs.

3.2 CGAN [3]

The Pix2Pix algorithm uses the general purpose architecture for image-to-image translation detailed in Isola et. al. It is composed of two pieces: the generator, and the discriminator. The loss functions for Pix2Pix is $\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$. The generator tries to minimize the function while the discrimator's goal is to maximize it. By adding an additional $L1$ loss function $\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1]$, the generator is not only incentivized to fool the discriminator, but also to try to output images closer to the ground truth.

We used $\lambda = 100$, patch size $N = 70$ and trained the model using Adam optimization algorithm with learning rate $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$ and $\epsilon = 10^{-7}$. We used transfer learning with the final checkpoint of pretrained Sketch2Shoes model provided in the repository as starting point and trained with total of 125 epochs and batch size of 1.

3.3 CycleGAN [7]

CycleGAN is built on the Pix2Pix image architecture. In this case, two PatchGAN discriminators $(D_X, D_Y$ discriminate between the images while two U-net generators (G, F) generate the images.The loss function is given by $\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_y[\log D_Y(y)] + \mathbb{E}_x[\log -D_Y(Gx))]$, with the discriminator and generator assuming the same objectives as before. In addition, a cycle consistency loss $\mathcal{L}_{CYC}(G, F) = \mathbb{E}[||F(G(x)) - x||]_1 + \mathbb{E}_y[||G(F(x)) - y||]_1$

We used $\lambda = 10$ and trained the model using Adam optimization algorithm with learning rate $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$ and $\epsilon = 10^{-7}$ with a total of 30 epochs with batch size of 1 using last checkpoint of CycleGAN on Canny edge detection sketches.

4 Results

4.1 Visual Results of MUNIT

Figure 3: Mixing the styles of different dresses to create new outputs

In the first epoch, the neural net learned how to incorporate the edges into its outputs, and generally left the empty areas of the image empty. The actual coloring was poor, and the images had various artifacts including discoloration and random spots of color in empty spaces.

During the next 10 epochs the model quickly learned how to color within the outline of the dresses, learning how to produce solid color dresses quite well and also learning to create interesting patterns. After this, progress began to plateau and image quality did not improve significantly aside from fewer artifacts and more realistic shading. As a result of the optimizer prioritizing realism in image generation, this also led to fewer interesting patterns in later epochs which might not be a desirable feature in a creative tool.

In general, MUNIT outputs tended to ignore the inner details of the sketches, instead generating its own pattern given the sketch outline. This is to be expected, since the MUNIT architecture learns to create diverse images by separating the content of the image from its style. As a result, the model learns to separate the inner patterns from the outline, since the patterns are considered a part of the dress style and outputs only images which share the same content, not the same style. The benefit of this is that MUNIT can produce truly random images given random style codes as opposed to other methods which can only produce more or less deterministic outputs. This also allows MUNIT to produce image to image translations

like that of CycleGAN, except there is no limit to how many modes that MUNIT can translate images to, since it learns a higher dimensional style space rather than discrete styles (Figure 4). Furthermore, the separation of the image generation process into content and style allows us to also specify the desired style. Given one sketch, we can generate images which emulate the style of any arbitrary dress we wish as shown in Figure 3.

4.2 Visual Results of CGAN

We trained CGAN multiple times with different hyperparameters such as batch size and number of epochs. We found that with transfer learning from Sketch2Shoes, training the model for 125 epoch yields the best FID score. Both visual results and FID score indicates that after around epoch 100, the model starts to converge and no significant improvement is observed afterward. The first few epochs showed that the model could quickly apply the pretrained weights on the new dataset by recognizing sketch edges and filling in major colors. By epoch 75, the model could generate sharper outlines for both dark-colored and bright-colored dresses. The model also learned to include the details on transparent fabrics at the bottom of dresses as well as the folds and creases (Figure 5). By epoch 100, the model managed to translate the intricate patterns from the sketches to realistic images with a consistent color distribution most of the time, shown in Figure 6. Although it is able to produce more diverse color schemes for dresses than CycleGAN (to be discussed in the next section), there is still room for improvement in the realism of contrasting colors.

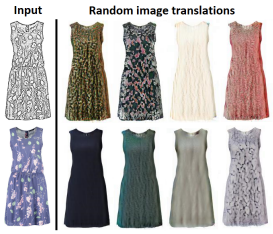

Figure 4: Eight random image translations of the same sketch generated by MUNIT

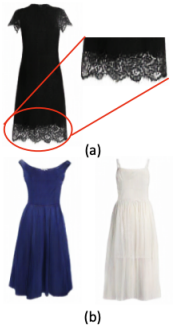

Figure 5: Examples of folds and creases on test set outputs of CGAN

Figure 6: Examples of CGAN test set outputs (left: model-generated, right: ground-truth)

4.3 Visual Results of CycleGAN

Figure 7: Examples of trade-off between diversity and quality during training

We continued training CycleGAN using the the last checkpoint of cycleGAN trained on Canny edge, but now with the processed CycleGAN-output sketches. As expected, during the first few epochs, the model was already able to capture the details quite well since it inherited the trained weights for these sketches. However, we also noticed that it struggled a lot with coloring the dresses. Since the sketches were now processed to be Grayscale, the model could no longer use color information to inform the generated colors (See section 2.3). After 15 epochs, it learned to produce different colored dresses although it still struggled to produce images with smooth coloring distribution and sharper outlines for the detailed patterns. We trained it for 15 more epochs and noticed that the quality of different fabric textures and complex patterns improved significantly. At the same time, it started to produce the same color scheme for most of the dresses that had detailed patterns. This behavior shows the trade-off between quality and diversity of images in CycleGAN. As CycleGAN is trained for more epochs, the image quality, especially realism, will continue to increase as the network converges on the best way to translate from sketch to dress. But in the process you will also lose out on diversity of colors and patterns (Figure 7). For our purposes, we need to strike a balance between realism and diversity so we decide to stop training after 30 epochs, which yields the lowest FID score.

In general, CGAN, CycleGAN, and MUNIT are able to learn to produce some basic features of dresses like color schemes, folds, shading, and creases. While MUNIT can't handle transparent or differently textured fabrics, CGAN and CycleGAN perform quite well in this regard. Compared to CGAN and MUNIT, CycleGAN produces much more realistic pictures, capturing most of the detailed patterns with impressively sharp resolution (Figure 9a). But, both CGAN and MUNIT outperform CycleGAN when it comes to having a more diverse color distribution (Figure 9ab). MUNIT generally

Figure 8: Examples of CycleGAN test set outputs (left: model-generated, right: ground-truth)

ignores or poorly translates the patterns of dresses (Figure 9ab). On the other hand, it does a good job in handling the lighting and smoothing effects for solid single-colored dresses to make them look more realistic (Figure 9c).

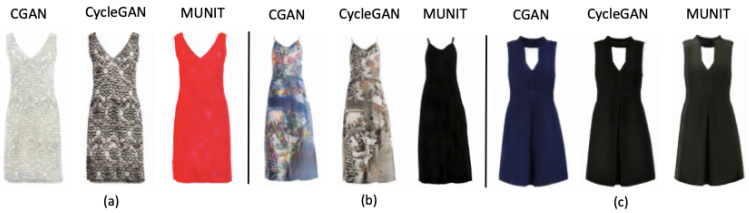

Figure 9: Examples of strengths and limitations for each model

4.4 FID score

In order to quantitatively estimate how similar our test set and ground truth images were, we calculated the Frechet Inception Distance (FID) [9] for Evaluating GANs. By calculating the covariance between real and generated image distribution, this score gives us an estimate of how close our test output and ground truth images are to each other. We run FID using a pre-trained Inception V3 network.

Edge Detection Model	CGAN	CycleGAN	MUNIT
HED	47.9	59.9	27.5
Canny	39	55.4	27.5
CycleGAN Sketches	22.7	20.1	24

Table 1: Comparing FID Scores for various edge detection algorithms and Models

The FID score for CGAN and CycleGAN indicates a significant improvement when we use CycleGAN sketches over HED and Canny edge detection algorithms. On the other hand, MUNIT's FID score does not improve much on CycleGAN sketches because based on our visual result, it tends to ignore the outlined patterns.

While the above score does give us a quantitative metric to determine how close the images produced by our model are to the ground truth and is helpful in evaluating cross-model performance, it is critical to holistically evaluate the visual quality on generated images under the lens of humans. Two main goals we wish to achieve are realism and diversity. The generated images should be realistic as well as having a diverse distribution, not limited to only one or two styles or colors.

4.5 Human Perceptual Study

In order to assess the quality of images (in particular how 'real' the images looked), we ran a small perceptual study that consisted of two phases. We collected data by creating a website that hosted the two phases of our study and sent participating invitations to our institutional peers. We divided the participants into three different groups A,B and C with no overlapping tasks which means one group is only tasked for a specific phase or model.

4.5.1 Phase 1

In the first phase, participants in group A were shown four images for unlimited time and were asked to judge which image looked the most realistic. The four images were outputs generated with respect to the the same input sketch. Out of the four images, one was generated by the CGAN model, one was generated by the CycleGAN while the remaining two were generated by MUNIT.

Model.	% Selected as *most realistic* $\pm$ Standard Error
CycleGAN	38.69% $\pm$ 1.17 %
MUNIT	29.16% $\pm$ 1.09 %
CGAN	32.14% $\pm$ 1.13 %

Table 2: Comparing results for Phase 1 of the Perceptual Study

The results of phase 1 show CycleGAN as the clear winner when it came to image realism. While a higher percentage of people did choose images generated by MUNIT than CGAN, after accounting for the fact that we include 2 MUNIT images in each sample, CGAN also outperforms MUNIT. This matches both our intuitions from the visual results and the results from evaluating FID score.

4.5.2 Phase 2

We selected the two models that performed the best in the first phase (in this case the selected models were CycleGAN and CGAN). Then, we showed participants in group B and C an image randomly chosen either from the model or ground-truth for only 1 second and asked them whether they thought the image was real or not. Note that each model has different set of participants.

Model.	% Selected as *real* $\pm$ Standard Error
Ground Truth	$61.87\% \pm 1.6\ \%$
CycleGAN	$45.41\% \pm 1.5\ \%$

Table 3.1: Comparing results for CycleGAN against ground truth

Model.	% Selected as *real* $\pm$ Standard Error
Ground Truth	$69.34\% \pm 1.4\ \%$
CGAN	$53.07\% \pm 1.3\ \%$

Table 3.2: Comparing results for CGAN against ground truth

From these results we can see that CycleGAN and CGAN both do very well at emulating real images. We are able to fool humans almost 50% of the time when humans were only $60 - 70\%$ accurate with real images..

To compare the performance between CycleGAN and CGAN for this phase, we do not explicitly compare the raw percentages selected as real for each model. This is because we need to take into account the different percentage selected as real for ground truth in each model's session, one with 61.87% and the other with 69.34%. Therefore, we compare how well each model performs given how well the ground-truth performs for each model's session. That means, the ground-truth did around 16.46% number of trials better than CycleGAN. On the other hand, ground-truth did approximately 16.27% number of trials better than CGAN. This indicates that both CycleGAN and CGAN have a similar performance in fooling humans with the fake images.

5 Conclusion

In conclusion we find that while the CGAN and CycleGAN models perform better in image realism than MUNIT, MUNIT as expected generates more diverse images. In general, we notice a tradeoff between image realism and diversity of outputs in all of three models which needs to be optimized for our specific task. In MUNIT, better image realism meant fewer interesting patterns were generated. In CGAN and CycleGAN, better image realism meant lower diversity in colors. This tradeoff is also reflected in the comparative underperformance of MUNIT in our human trials. We could potentially overcome this tradeoff by expanding the dataset to include more diverse images and correcting for biases, or by optimizing loss function weights and hyperparameters. Further work needs to be done in this regard.

We also conclude that for the task of generating realistic images from design sketches the most effective method of generating training data "sketches" is not the edge detection algorithms frequently used in prior projects like Edges2Shoes or Edges2Cats. Instead, we recommend the use of algorithms like the CycleGAN model which can generate sketches with better details and shading. Not only do these produce far better outputs, but also they are a far better representation of real fashion design sketches. Further work also needs to be done in this regard. In particular, we believe that since many real design sketches already include color, more experiments should be done on RGB sketches rather than Grayscale. As we have seen in section 2.3 this could lead to greatly improved output realism. Actual sketches from real designers should also be incorporated into the dataset, especially in the test set, in order to evaluate our tool's actual usefulness to designers.

6 Code

The code that we implemented for Pix2Pix&CycleGAN as well edge detection algorithms can be found at `https://github.com/vythaihn/Sketch2Fashion-pytorch-CycleGAN-and-pix2pix`
The code that we implemented for MUNIT can be found at `https://github.com/Manya-bansal/MUNIT/workingbranch`
The code that we implemented for calculating FID sore can be found at `https://colab.research.google.com/drive/1igspdz0bXm8ZXhsDeyNjy6QzD0DLF-ED?usp=sharing` This code was taken from `https://github.com/mseitzer/pytorch-fid`
The code that we implemented to build a website for Human Perceptual Study can be found at `https://github.com/vythaihn/Skech2Fashion`

7 Contributions

- Data processing using HED: Vy, Manya
- Data processing using Canny: Vy
- Data processing on CycleGAN edges: David
- Pix2Pix training/testing: Vy, David
- CycleGAN training/testing on the different edges: Vy
- MUNIT training/testing on the different edges: David, Manya
- FID research and testing: Manya, Vy
- Human Perceptual study (creating website): Vy, Manya
- Human perceptual study (data analysis): Manya
- Writing the report framework: Manya, David

References

[1] Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). *Image Style Transfer Using Convolutional Neural Networks*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference On, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

[2] Lefakis, L., Akbik, A., Vollgraf, R. (2018). FEIDEGGER: A Multi-modal Corpus of Fashion Images and Descriptions in German. LREC.

[3] Isola, Phillip Zhu, Jun-Yan Zhou, Tinghui Efros, Alexei. (2017). Image-to-Image Translation with Conditional Adversarial Networks. 5967-5976. 10.1109/CVPR.2017.632.

[4] Huang, Xun, et al. "Multimodal Unsupervised Image-to-Image Translation." ArXiv:1804.04732 [Cs, Stat], Aug. 2018. arXiv.org, http://arxiv.org/abs/1804.04732

[5] Xie, Saining, and Zhuowen Tu (2015). "Holistically-Nested Edge Detection." ArXiv:1504.06375 [Cs], arXiv.org, http://arxiv.org/abs/1504.06375.

[6] Canny, J. (1986). A Computational Approach To Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698.

[7]Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV).

[8] Hesse, Christopher, CGAN Tensorflow, (2017). GitHub repository, https://github.com/affinelayer/CGAN-tensorflow

[9] Heusel, Martin, et al. (2018). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." ArXiv:1706.08500 [Cs, Stat]. arXiv.org, http://arxiv.org/abs/1706.08500.