Using GANs and Encoders/Decoders for Image-to-Sketch Translation

15.S04: Hands on Deep Learning
Jan Reig Torra, Shurui Cao, Xinyao Han, Daniel Chung

1. Introduction and Problem Statement

There is a delicate balance between visibility and privacy, especially online where malicious agents take advantage of facial photographs for adversarial purposes. How can users accurately represent their appearance while preserving their privacy? And how can this process be done in an efficient and convenient way?

Face sketch synthesis is an exciting problem in computer vision and deep learning that involves image-to-image translation. It can generate sketches of human faces from input images, essentially translating a real photograph into a synthetic sketch that retains the photo's likeness. We are thus interested in image-to-image translation because it could automate avatar illustration for online user profiles, preserving user likeness in a way that conceals their photograph from the public.

To this end, our technical objective is to develop a deep learning-based model that can accurately and efficiently generate sketches of human faces. This necessitates the design and training of an Autoencoder or convolutional neural network (CNN) to capture the key features of a face through an encoder and generate a corresponding sketch through a decoder. The models need to be trained on a dataset of images of human faces and corresponding sketches and fine-tuned to optimize their performance.

The business impact of such a solution is the introduction of a novel alternative to traditional profile pictures in social networks. By using the generated sketch as a profile picture, users can maintain their privacy while still presenting themselves in a visually appealing way. This could be particularly valuable for individuals who prioritize privacy and security in their use of social media.

Our project explores the feasibility of using these generated sketches as profile pictures and developing a user- friendly interface for users to upload their own images and generate sketches. Our project has the potential to provide new and exciting business opportunities for social media companies by offering users a unique and engaging way to represent themselves online. By providing an innovative solution to the challenges of online privacy, we can help social media companies attract new users and retain existing ones.

2. Dataset

Our data consists of 188 color photographs of human faces and 188 pencil sketches corresponding to each face. The sketches were all created in the same artistic style, and each was made

from the same frontal angle. These image-sketch pairs were sourced from the Chinese University of Hong Kong and can be found on Kaggle at the link under references [1].

To standardize the images, we rescaled our data to a resolution of 256x256, meaning both the facial photographs and sketches were represented as 3-channel RGB arrays of shape (256, 256, 3). Each image was also normalized.F1

Further, we applied several data augmentations to both the train and test images in order to robustify the model against unforeseen variations in new images. Each image was flipped horizontally and vertically twice, rotated clockwise, flipped in the rotated position, rotated anti-clockwise, and flipped in that rotated position. This increased the amount of photo-sketch pairs in our dataset by a factor of 8, meaning that our augmented dataset consists of 1504 image-sketch pairs.

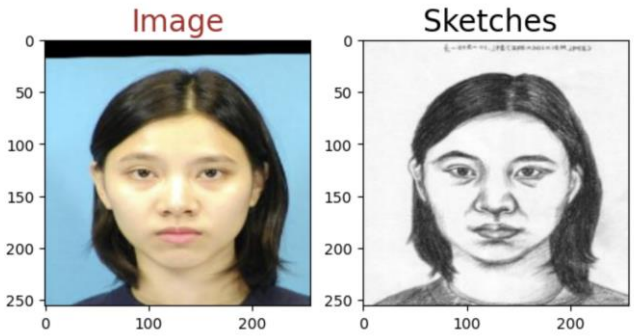


Figure 1: example of a photograph-sketch pair from the Chinese University of Hong Kong dataset

3. Methodology

3.1. Encoder/Decoder

The encoder component of the architecture is responsible for extracting meaningful features from the input image, creating a compressed representation of the image's content. This is typically achieved using a series of convolutional layers that capture hierarchical patterns and structures in the image. The decoder, on the other hand, takes this compressed representation and reconstructs a new image based on the extracted features. The decoder often employs a series of upsampling and convolutional layers to gradually increase the spatial resolution of the output image while maintaining its semantic information. By training the encoder-decoder model on pairs of related images (e.g., photographs and corresponding sketches), our model learns to transform input real human face images into their corresponding sketches, effectively performing image-to-image translation tasks.

For our project, we've defined two main functions. The first one is a downsampling function which contains a convolutional layer to take image inputs and process it to produce a smaller output image with the specified number of output channels. We can also specify the number of filters, kernel

size, and whether we apply batch normalization or dropout. The second function is an upsampling function which contains a convolutional layer to take the encoder output and to increase its size while reconstructing the essential features of the original image.

For our model, we generally pass our input images through multiple downsampling layers first and then pass the encoder output generated from downsampling into upsampling layers to get the sketch output. We've tried different combinations of parameters to find the best model, including the number of layers, the number of filters, kernel size, dropout, and loss function.

3.2. cycleGAN

GAN stands for Generative Adversarial Networks, which is widely used for generating new, and previously unseen data samples. It has a broad application for image generation. GAN consists of a generator and a discriminator. The generator encodes images to high-level representations and generates synthetic images. The discriminator takes the real and synthetic images as inputs, and evaluates the authenticity of the generated data. The generator and discriminator are trained simultaneously through adversarial training. Figure X shows the basic structure of the network.

For our project, we used a modified version of GAN, the cycleGAN, which doesn't need paired images for learning [2]. It focuses on unpaired image-to-image translation, where it learns to relate two different domains, the image and the sketch. It's composed of 2 GANs, 2 generators and 2 discriminators. One GAN is responsible for generating images from sketches, while the other GAN generates sketches from images.

The generators take input from one domain and generate an output in the other domain. The discriminators then evaluate the authenticity of the generated data. The cycleGAN framework is trained through adversarial training, where the generator tries to generate output that can fool the discriminator, while the discriminator tries to distinguish between real and fake images. This process is repeated until the generator produces output that is indistinguishable from real data, and the discriminator can no longer identify the generated data as fake.

We use the cycleGAN structure from the paper *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* for this project [2].

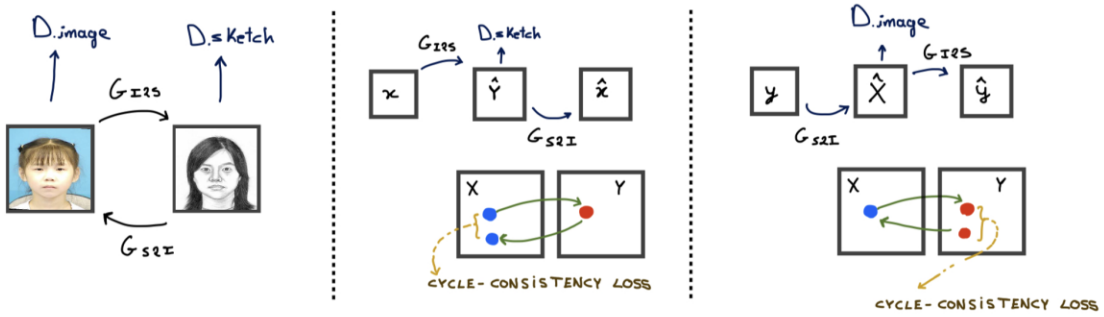


Figure 2: cycleGAN structure

From Figure 2, X and Y represent image and sketch respectively. GI2S and GS2I are generators that map images from one domain to another (e.g., GI2S: image to sketch, GS2I: sketch to image). D_image and D_sketch are associated discriminators. D_sketch encourages GI2S to translate images into synthetic outputs that are indistinguishable from real sketch, and vice versa for D_image and GS2I. We use adversarial loss for the above process.

As we can see in Figure 2, cycle consistency loss is a key component of the CycleGAN architecture that enforces the constraint that if an image is translated from one domain to another and then back again, it should closely resemble the original image. This constraint helps to ensure that the generated images are not only realistic but also preserve important features and details of the original images. In CycleGAN, cycle consistency loss is calculated in both the forward and backward directions. This is done by calculating the difference between the original image and the reconstructed image after passing through both generators.

4. Results

4.1 Encoder/Decoder

The best model that we found through parameter tuning achieves 71.5% accuracy on the test set. Its encoder uses regularization with lambda=0.5 and its decoder uses regularization with lambda=0.3. This also involved using a kernel size of 5 and performing gradient descent based on mean absolute loss. Because training for 3 epochs initially produced poor results, we trained our best model for 2000. Training accuracy, encouragingly, made a steady climb and was even surpassed by testing accuracy. A link to our implementation and training of the encoding / decoding approach can be found under references [5]

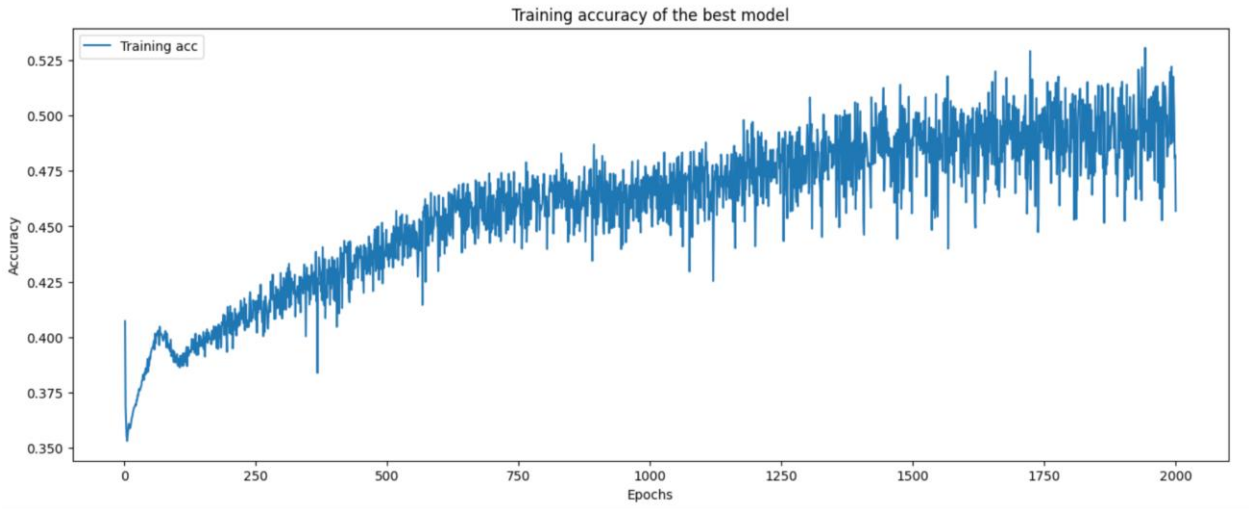


Figure 3: Training accuracy per epoch for our best model

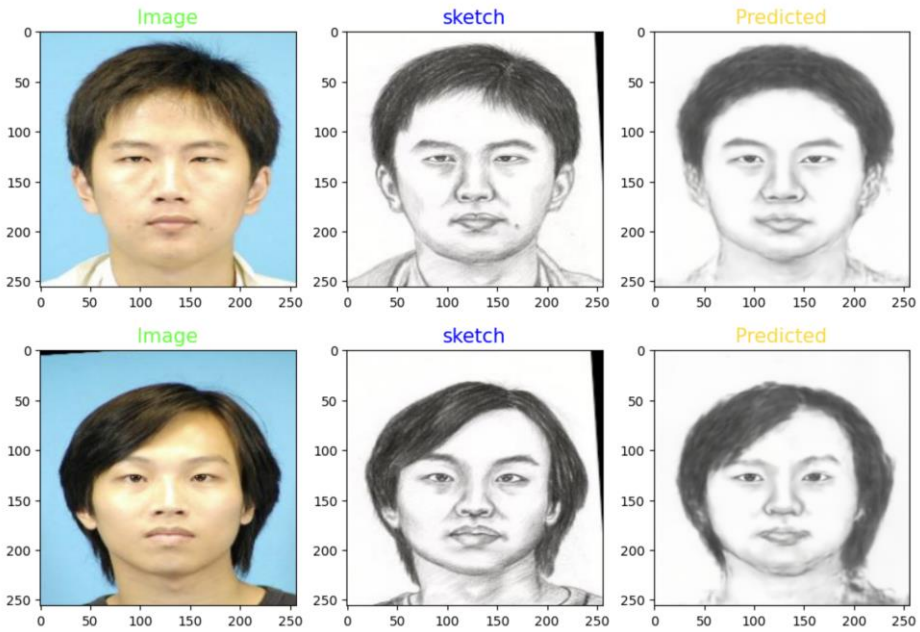


Figure 4: Image to sketch translation results on the original dataset with our best model

We can see that the predicted sketches from our model have high resemblance to the original sketches. While the details are lost in the predicted sketches, the model is able to reconstruct the original human faces clearly.

We also tried applying this model to our photos, which are not in the original dataset. However, the predictions are not as high quality as those on the original dataset. Below is the visualization of the best model's prediction based on our photos.

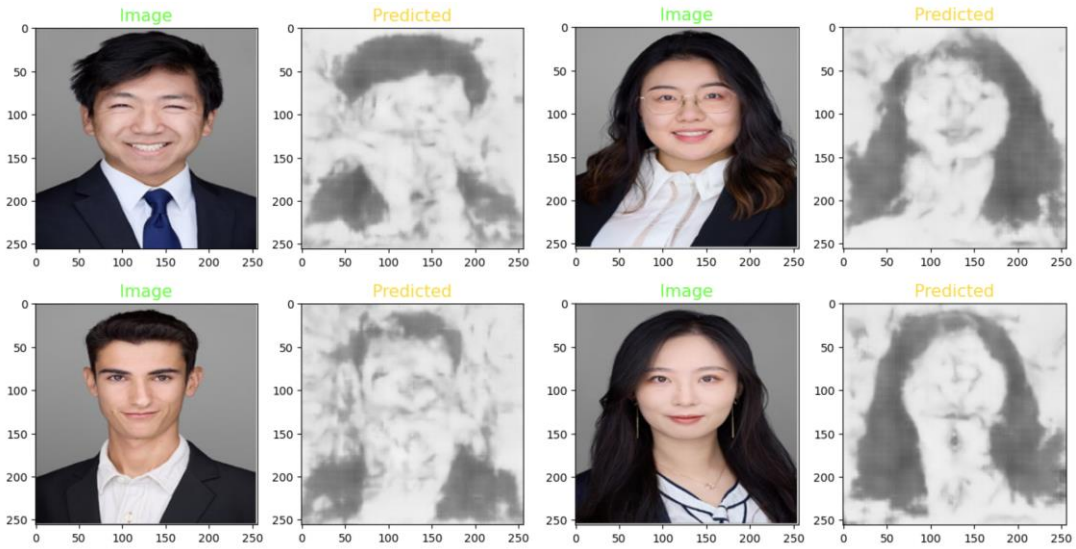


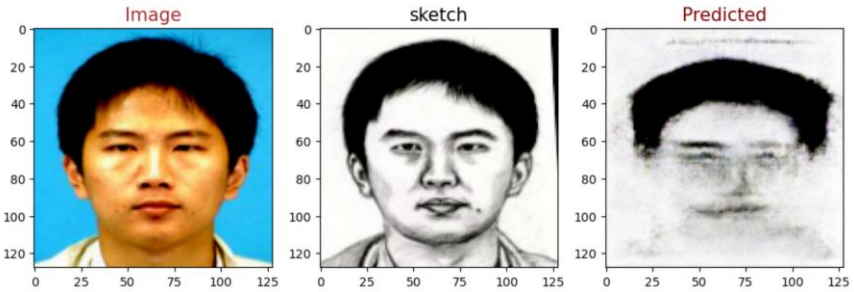
Figure 5: Image to sketch translation results on our photos with our best model

We can see that the best model captures the most obvious features of the photos, which is the area with the darkest colors like hair and clothes. While the model resembles the shape of hair and clothes roughly, it fails to capture the details in the face area. We'll discuss the possible reasons in the next section.

4.2 cycleGAN

We used the pre-trained weights on similar image-to-sketch tasks and fine-tuned the GAN on our dataset with 2000 epochs. The code for our cycleGAN implementation can be found under references [5]

Figure 6 contains the output sketches of the images that were tested in both the autoencoder and the cycleGAN. As depicted, the performance of the cycleGAN is inferior to that of the autoencoder.



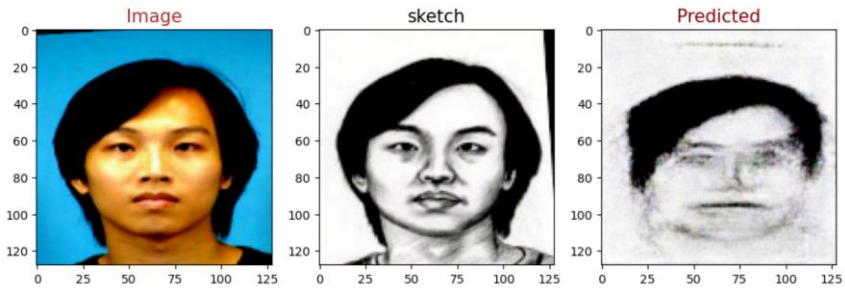


Figure 6: Image to sketch translation results on the original dataset with cycleGAN

We visualize the image to sketch translation results here:

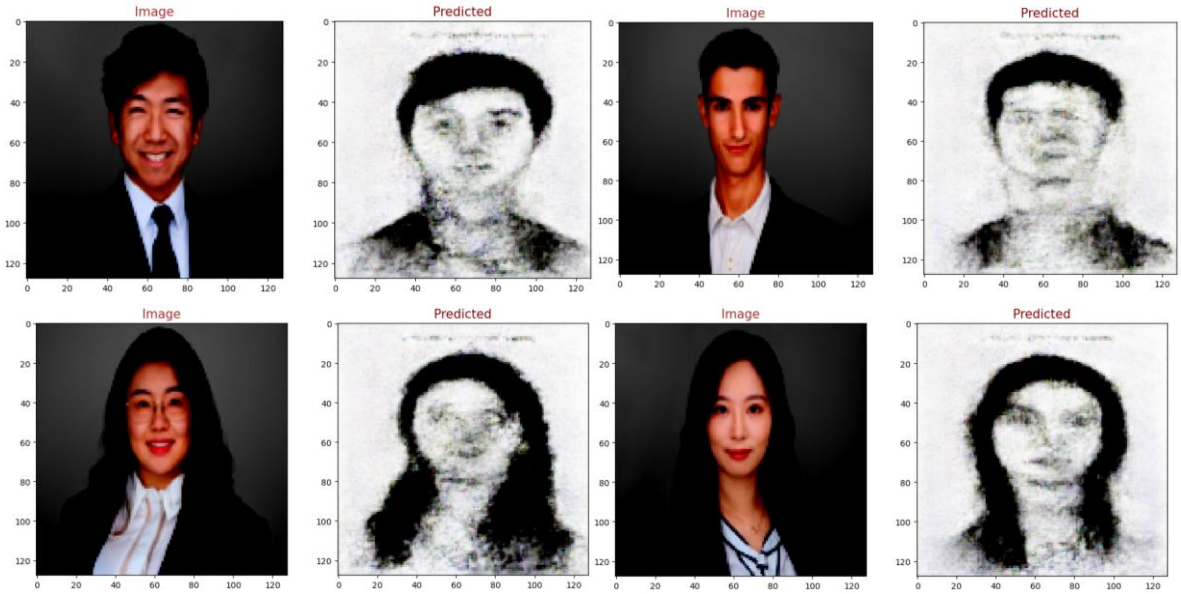


Figure 7: visualization image to sketch translation results with cycleGAN

From the visualization, we can see that these sketches are markedly more blurred than the training ones. This is an observation we discuss further in section 5.2.

5. Discussion and lessons learned

5.1 Encoders/Decoders

One primary reason that our models perform poorly on our photos is overfitting. They learn to recognize specific features or patterns in the original training images but fail to generalize well to new images. The models may be too complex, capturing noise and peculiarities in the training data instead of learning the underlying structure.

Another reason for poor performance on new data could be the lack of diversity in the training dataset. Our training dataset has limited variations in terms of image styles, lighting conditions, or object types, the model might struggle to handle different conditions or objects that were not present in the

training data. For example, our photos have gray backgrounds while the original dataset all use blue backgrounds. A dataset with more diverse images and sketch styles might be able to improve the performance of our models.

5.2 cycleGAN

When we train a cycleGAN model for image-to-sketch tasks, we also have the reverse translation from sketch-to-image because we are training 2 generators and 2 discriminators. We also use the F generator from sketch to image to translate sketches back to images based on previous work done in this area [3]. The visualization of our model visualization can be seen below. Similar to the image-to-sketch translation task, the image here is also blurred.

Figure 8: sketch to image translation with cycleGAN

Some of the possible reasons are insufficient training data and training time. Our dataset contains only about 150 images and sketches and lacks diversity in sketches, therefore the generators might not learn to create sharp and realistic images as expected. We also limit the number of iterations to 2000 because of capacity and computing power constraints. We assume the model is not trained for a sufficient amount of time and thus may not have converged to the optimal solution.

Some possible solutions would be to increase the size of the dataset, augment the dataset with more diverse examples, increase the number of iterations and adjust the learning rate. We would work on them if time permits.

5.3 Final remarks

In conclusion, this project on image to sketch translation using an Autoencoder and cycleGAN has been an interesting learning experience. We have gained valuable insights into the workings of complex convolutional neural network architectures and their application in computer vision.

One of the main takeaways from this project is the importance of understanding the architecture of a model when experimenting with different CNN and parameters to achieve better accuracy. We learned that there are various approaches to improving the performance of these models, such as adjusting hyperparameters, modifying the architecture, or using pre-trained models. Additionally, we found that getting diverse training data is crucial for achieving good results.

However, we also recognized that there are limitations to the accuracy that can be achieved with these models, particularly when it comes to data availability, time, and computational resources. We realized that training these models can be computationally expensive and time-consuming.

6. References

1. https://www.kaggle.com/code/theblackmamba31/photo-to- sketch-using-autoencoder
2. https://github.com/Harshpatel44/Image-to-Sketch-using-Cycle-GAN
3. Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks." 1703.10593.
4. https://drive.google.com/file/d/1PZ0h9y-TINmuoFa1i5lLlWqMEyXK263N/view?usp=sharing
5. https://colab.research.google.com/drive/1hmeyFn24AwUa733nXGWHU_teMIAThNQq?usp=sharing