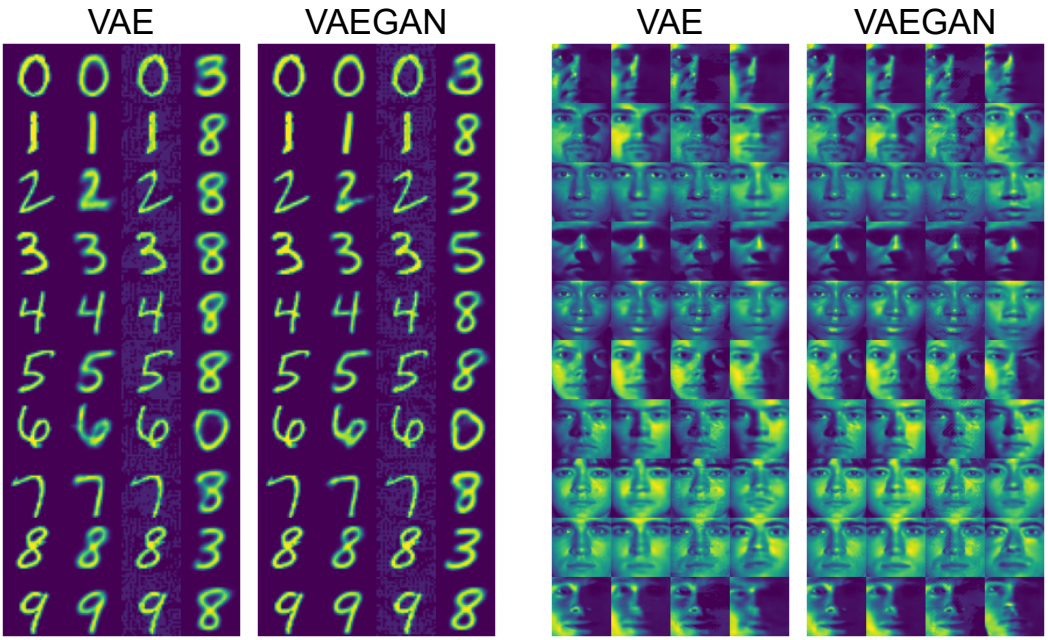
A ARCHITECTURES AND PREPROCESSING DETAILS FOR THE EXPERIMENTS

For MNIST, we used the architecture from (Chollet) for the targeted VAE and the same architecture with a discriminator from DCGAN (Radford et al., 2015) for the targeted VAEGAN. Further, we applied the architectures of DCGAN (Radford et al., 2015) for the adversary's generator and discriminator. Note that the encoder part of the generator is the inverse of the decoder part. For the adversary's classifier, we adopted the architecture in (Jang). First, we created our targeted models by training the VAE and VAEGAN with the training samples, 20 latent space, 250 batch size, Adam optimizer (Kingma & Ba, 2014) and 100 epochs. Second, we pretrained the adversary's classifier with the same setting as the targets.

For CYFD, we utilized the architecture from (Stewart) to create the targeted VAE and the same architecture with a discriminator also from (Stewart) to create the targeted VAEGAN. Note that the work in (Stewart) is for generating images of the CelebA dataset (Liu et al., 2015) which is also a human-face dataset. Moreover, we used the architectures in (Stewart) to create the adversary's generator and discriminator. Note that the encoder part of the generator is the inverse of the decoder part. Then, we applied the architecture of the MNIST classifier to create the adversary's classifier. After that, we trained the VAE and VAEGAN with the training samples, 80 latent space, 245 batch size, Adam optimizer (Kingma & Ba, 2014) and 100 epochs. At last, we pretrained the classifier with the same setting.

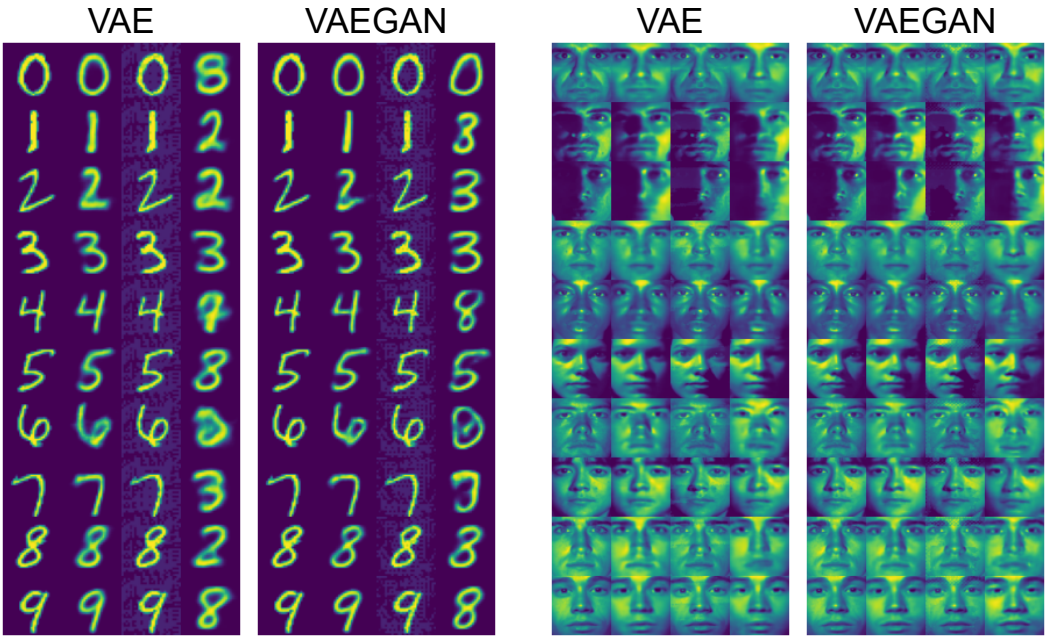
B FURTHER EXPERIMENTS ON THE UNTARGETED ATTACK



(a) MNIST (b) CYFD

Figure 3: The result of our standard framework on the left is from the VAE and on the right is from the VAEGAN. The images 1) in the first column are clean; 2) in the second column are the reconstructed images for the images in the first column; 3) in the third column are adversarial examples concerning the images in the first column; 4) in the last column are the reconstructed images for the adversarial examples.

Figure 3a and 3b show the results after evaluating the VAE and VAEGAN on the test samples of MNIST and CYFD with the standard framework. Note that the first two columns show the clean inputs and their reconstructed images of the models. The last two columns show the adversarial examples and their reconstructed images of the models. Noticeably, our generator can find weaknesses on both the VAE and VAEGAN.



(a) MNIST (b) CYFD

Figure 4: The result of our universal-attack framework on the left is from the VAE and on the right is from the VAEGAN. The images 1) in the first column are clean; 2) in the second column are the reconstructed images for the images in the first column; 3) in the third column are adversarial examples concerning the images in the first column; 4) in the last column are the reconstructed images for the adversarial examples.

Figure 4 shows that perturbation generated from this framework is less effective than the standard one when we compare it to Figure 3. Furthermore, we also found that this framework was less effective than the standard one in the targeted attack. However, the universal-attack generator can perform better than the standard generator in real-time because it can prepare perturbation beforehand, regardless of the input.

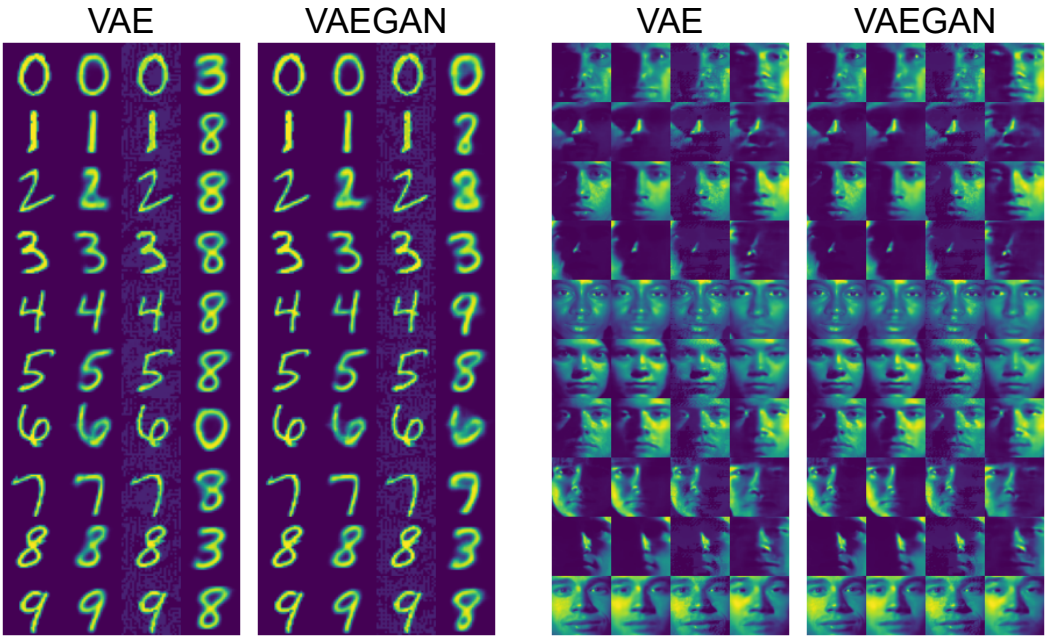
C Comparison between adversarial examples generated from the standard framework and the one without the discriminator

Figure 5 shows the results from the standard framework without the discriminator. Noticeably, even though the adversarial examples look valid, the adversarial examples from the standard framework with all the components in Figure 3 are more valid then the ones from this framework without the discriminator.

D Further experiments on the targeted attacks

Figure 6 demonstrates the results from our generators attacking the VAE and VAEGAN. Note that the images on the left are adversarial examples generated by our generators from test samples, and the images on the right are their corresponding reconstructed images. Further, the images in column i belong to class i, and row i is the targeted class i. Noticeably, our generators can successfully find perturbation to change the class of the reconstructed image to the desired class. Some classes are difficult to alter to other classes (e.g., class 0 and 6).

Further, Figure 7 shows the results from our generators on the VAE and VAEGAN in CYFD. Note that the images on the left are the same in the top and bottom since they are from the targeted classes (i.e., 2, 10 and 28 respectively). The images in the middle are adversarial examples created by our



(a) MNIST (b) CYFD

Figure 5: The result of our standard framework without the discriminator on the left is from the VAE and on the right is from the VAEGAN. The images 1) in the first column are clean; 2) in the second column are the reconstructed images for the images in the first column; 3) in the third column are adversarial examples concerning the images in the first column; 4) in the last column are the reconstructed images for the adversarial examples.

generators and originally from class 0, 3, 6 and 14, respectively. Furthermore, the images on the right are their corresponding reconstructed images. Explicitly, our generator on the targeted class 2 can make the reconstructed images of class-0 and class-14 images similar to an image from class 2. Also, the targeted class 10's generator can make the reconstructed images' eyes of class-0, class-3 and class-6 images similar to those from class 10. The generator on the targeted class 28 is also effective on images from class 0, 3 and 6.

Table 8 shows the success rates of our generators on each targeted class with samples of each class in MNIST on VAE. Note that we ignore the results of our generators on the targeted classes that are the same as tests samples'. Also, the bold success rates are higher than or equal to 80%. We found that the results on VAEGAN were similar to this one, as seen in Table 9. However, there are only some different cases (e.g., the targeted class 3 with class-2 samples and the targeted class 0 with class-5 samples).

Further, these results imply that class-1 samples are easy to find adversarial examples due to the high success rates on the VAE and VAEGAN. Moreover, class-0 samples are difficult to find adversarial examples because the success rates are not higher than 80% for any targeted class. Additionally, the generators on the targeted class 8 and 9 are the most successful attacks because they can achieve high success rates with several classes.

E CODE TO REPRODUCE EXPERIMENTS

Since the code may be misused in a malicious way, we can share the code when we receive a request from a formal institution.

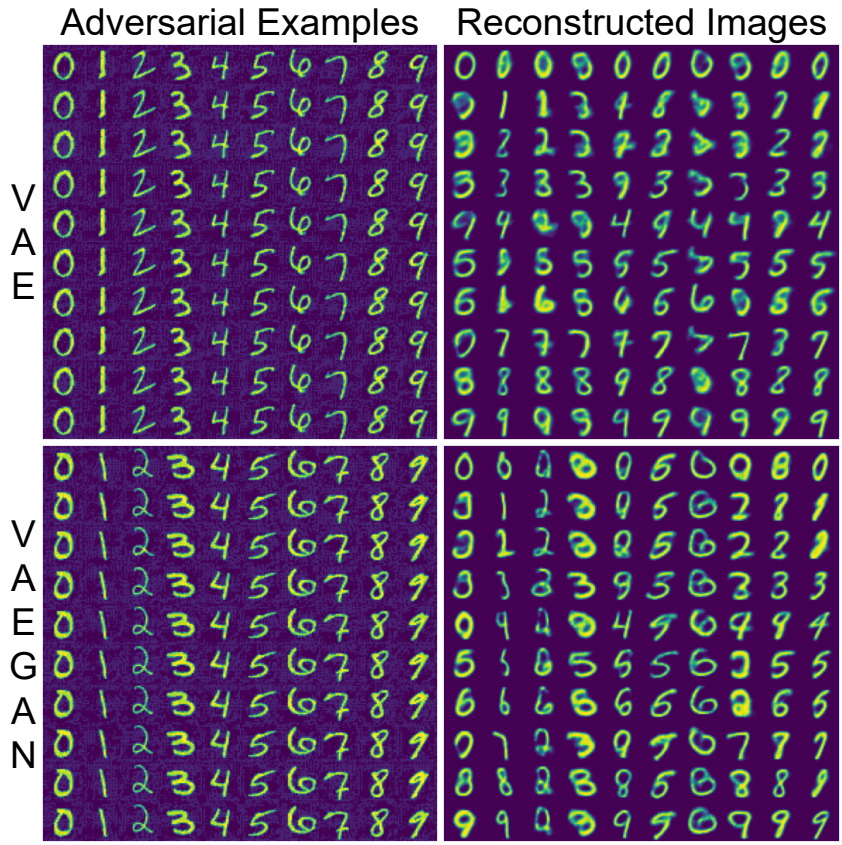


Figure 6: Adversarial examples (left) and their corresponding reconstructed images (right) of the VAE (top) and VAEGAN (bottom). The columns represent images from class 0-9 respectively. The rows represent the targeted class 0-9 respectively.

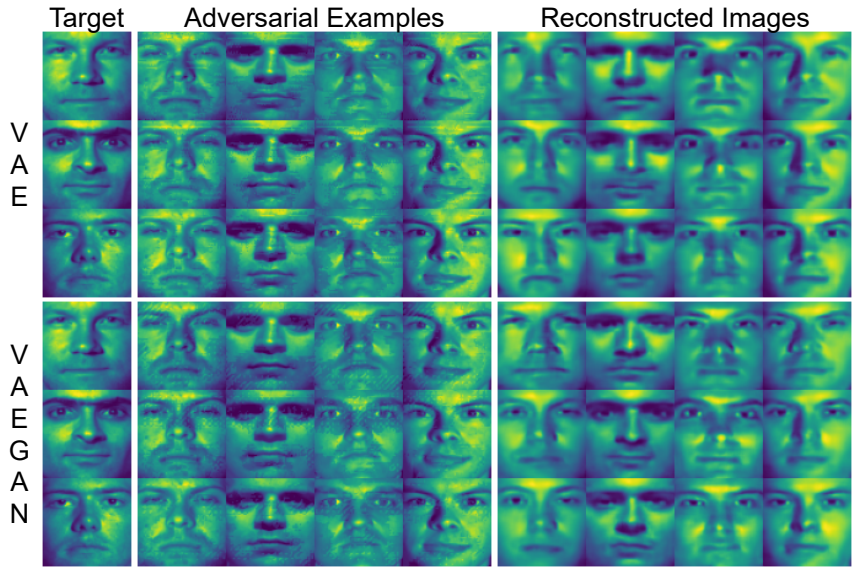


Figure 7: Targeted-class images (left), adversarial examples (middle) and their corresponding reconstructed images (right) of the VAE (top) and VAEGAN (bottom) in CYFD

Table 8: Success rates of our generators with the targeted classes (rows) on the VAE with samples from each class (columns) in MNIST

	0	1	2	3	4	5	6	7	8	9
0	100%	**85%**	35%	64%	**91%**	**89%**	**98%**	44%	63%	**95%**
1	2%	100%	12%	8%	55%	23%	47%	27%	33%	53%
2	71%	**92%**	100%	**86%**	78%	65%	**92%**	76%	**93%**	**82%**
3	67%	**82%**	54%	100%	42%	**93%**	76%	**84%**	**92%**	**89%**
4	30%	**96%**	11%	45%	100%	19%	**85%**	65%	20%	**91%**
5	49%	49%	3%	59%	35%	100%	74%	27%	59%	**86%**
6	37%	78%	13%	15%	61%	52%	100%	67%	23%	38%
7	70%	**97%**	55%	70%	**95%**	61%	57%	100%	**87%**	**97%**
8	64%	**94%**	67%	**86%**	**86%**	**97%**	79%	**88%**	100%	**95%**
9	71%	**95%**	18%	61%	**98%**	**88%**	**87%**	**82%**	**86%**	100%

Table 9: Success rates of our generators with the targeted classes (rows) on the VAEGAN with samples from each class (columns) in MNIST

	0	1	2	3	4	5	6	7	8	9
0	100%	**91%**	57%	52%	**87%**	62%	**92%**	**84%**	75%	**95%**
1	2%	100%	28%	26%	38%	37%	26%	54%	36%	55%
2	65%	**89%**	100%	57%	79%	33%	**82%**	**82%**	**86%**	75%
3	65%	**90%**	**83%**	100%	53%	**94%**	79%	**83%**	**89%**	76%
4	16%	**96%**	32%	8%	100%	27%	82%	59%	58%	**89%**
5	39%	**91%**	22%	**87%**	66%	100%	**84%**	43%	**83%**	75%
6	62%	**91%**	42%	29%	**95%**	**83%**	100%	23%	79%	**84%**
7	72%	**97%**	60%	52%	71%	50%	25%	100%	68%	**96%**
8	69%	**91%**	75%	77%	**92%**	**86%**	**80%**	**88%**	100%	**93%**
9	47%	**92%**	32%	39%	**98%**	63%	69%	**95%**	**83%**	96%