Sketch to Image Translation using GANs

Lisa Fan, Jason Krone, Sam Woolf
Tufts University, MA, United States
{Lisa.Fan, Jason.Krone, Samuel.Woolf}@tufts.edu

Abstract

In this work we explore the effect of using a discriminator with 2N output classes (real and fake scores for each target class) as well as different types of loss functions on the quality of images generated using an image-to-image cGAN (Conditional Generative Adversarial Neural Network). Specifically, we experiment with two different loss functions, the first of which is a fairly standard cross entropy loss (we call this the 2N loss) and the second attempts to take advantage of extra information provided by our 2N classification scheme (we call this the penalty loss). We find GANs trained using our 2N loss and penalty loss produce images that are of similar if not better quality than the standard GAN loss.

1. Introduction

Image generation is a valuable tool in the computer vision field as well as the world outside it. For example, image generation can be used in unsupervised contexts to generate training images for sparse categories. Outside of the field, image generation has many artistic use cases. Existing work, such as iGAN [9], has shown the success of using Generative Adversarial Networks (GANs) to create artwork. In this paper we further explore the use of GANs to create artwork by applying existing image to image translation techniques to generate photos from sketches.

The traditional implementations of GANs utilize discriminators that solely output a measurement of the real/fake quality, or realness, of an input image. These metrics seem to work in most contexts. However, in a few contexts, the input images have the potential to be coupled with additional information, including aspects such as image class or category. This paper explores the augmentation of traditional GAN discriminators in order to produce outputs that can utilize this extra information. In theory, when a discriminator can make inferences into the class of an image, one can implement a nuanced loss function that incorporates this extra information. Then, both the discriminator and generator can be more effectively updated based off of this additional information.

In previous works, discriminators have been supplemented to produce output vectors of length N+1 instead of the traditional realness output. This N+1 vector gives information on class, as well as realness. Our paper presents an additional improvement that furthers this idea. We explore the concept of a discriminator outputting a vector of length 2N, where the first N entries correspond to the image classes of real images, and the second N entries refer to the image classes of generated images. As we show, this additional information allows for the discriminator and generator to be more efficiently updated, which in turn leads to better results.

2. Background & Related Work

As we developed a novel strategy for generating images using cGANS, we were heavily influenced by existing work. Specifically, the concept and loss function for a multi-class GAN discriminator is based off of ideas presented in "Improved Techniques for Training GANs" [6]. Our core network architecture is based upon a network presented in "Image-to-Image Translation with Conditional Adversarial Networks" [2].

In the original GAN model proposed by Ian Goodfellow, the GAN discriminator has a single probabilistic output, and attempts to decipher whether an input image is real or generated. Then, a simple loss based off of this sole value is used to update both the generator and discriminator [1]. As first proposed in "Improved Techniques" [6], the discriminator can be restructured so that it can give us more information than just a simple probability. The authors suggest creating a discriminator that has output of N+1 values, where N is the number of classes in the training data set. In this output, the first N values correspond to the classes, and the N+1^{th} value corresponds to any generated image. By creating a discriminator that outputs class information instead of the probabilistic realness, one is able to calculate a better informed loss, and thus better refine both the generator and discriminator. In this paper, we take the concept one step further, attempting to create a more expressive discriminator by increasing the number of categories in the output of

the discriminator. Our discriminator now has an output of 2N classes: N classes for real images, and N classes for generated images. Our hypothesis is that by utilizing this extra information, our network will calculate a more nuanced loss, and then be able to more efficiently improve both the generator and the discriminator, leading to a more effective GAN.

The "Image-to-Image" paper [2] presents a novel way to train a conditional GAN as a solution for image-to-image translation. Specifically, the network uses a generator to first encode an image to a high-level representation, and subsequently decode the representation into a generated image. By training the cGAN on input-output image pairs, one can train the generator to create images that are, in theory, indistinguishable from the given output population. Our novel approach relies heavily on the U-Net architecture of the generator and the convolutional layer architecture of the discriminator proposed in this paper.

3. Approach

3.1. Architecture

We use a pre-existing GAN implementation provided by the authors of "Image-to-Image" [2] as the basis for our model. The generator has two components: an encoder component, which takes the given sketch s and downsamples it to create a lower dimensional representation $\phi(s)$, and a decoder layer, which takes a vector containing $\phi(s)$ and produces an image. The generator contains skip connections between the i^{th} layer of the decoder and layer 8 - i of the encoder. The architecture for the generator is as follows:

- Encoder:
  C64-C128-C256-C512-C512-C512-C512-C512
- Decoder:
  CD512-CD512-CD512-C512-C512-C256-C128-C64

where C stands for convolution and CD stands for a deconvolution. All of the convolutions use 4x4 spatial filters applied with stride 2. Convolutions in the encoder downsample by a factor of 2 and in the decoder convolutions upsample by a factor of 2. Leaky ReLU activation functions with a leak of 0.2 are used between layers in the encoder and standard ReLU activations are used between layers in the decoder.

All of the convolutions in the discriminator use 4x4 spatial filters with a stride of 2 except for the final layer, which uses a stride of 1. Leaky ReLU activations with a leak of 0.2 are used in between the convolutional layers. And both the generator and discriminator are trained using the Adam update rule [3] with a learning rate of 0.0002 and momentum of 0.5. The discriminator produces a 30x30 output that corresponds to the realness of different patches of the input image. This out-of-the-box implementation was used as a baseline network to compare against our own models.

We modify the "Image-to-Image" [2] discriminator described above by adding a fully-connected layer to the end of the network, which outputs a 2N-dimensional vector of logits. By including N fake classes in our output, rather than a single fake class as described in "Improved Techniques" [6], we increase the discriminator's power to learn lower level features that differentiate between fake images of different objects. In contrast, having only a single class that represents fake images of all object categories forces the discriminator to look for high level features shared by all generated images that indicate an image is fake. Our discriminator has the following architecture:

C128-C256-C512-C1-FC125

The 2N-dimensional output vector has the form:

$$\text{output} = [l_{1R}, \ldots, l_{NR}, l_{1F}, \ldots, l_{NF}]$$

where R denotes a real object class, F denotes a fake object class, and N is the number of classes in our dataset. In this formulation, a class represents a real or fake photo of a particular type of object.

These logits can be turned into class probabilities using a softmax:

$$p_{model}(y = j|x) = \frac{\exp(l_j)}{\sum_{i=1}^{2N} \exp(l_i)}$$

We use these class probabilities to calculate our 2N loss and penalty loss, which we describe in the following sections.

3.2. 2N Cross Entropy Loss

We will first discuss the 2N cross entropy loss function, which is the simpler of the two loss functions used in our experiments. This 2N cross entropy loss function is inspired by the supervised component of the N+1 loss function outlined in the introduction and proposed in "Improved Techniques" [6]. The discriminator loss L_D contains two terms. The first term is a cross entropy loss for a real image x and sketch s pair taken from our training data distribution p_{data} with ground truth class y and target class y. The second term is a cross entropy loss for the image $G(s)$ generated from a sketch s with ground truth class y and target class y. The loss L_D is described by the following equation:

$$\begin{aligned} L_D = -(&\mathbb{E}_{x,s,y\sim p_{data}(x,s,y)}[\log p_{model}(y|x, s, y \leq N)] \\ &+ \mathbb{E}_{s,y\sim p_{data}(s,y)}[\log p_{model}(y|G(s), s, N < y \leq 2N)]) \end{aligned} \tag{1}$$

Similarly, the generator loss L_G contains two terms. The first term is a cross entropy loss for the image $G(s)$ generated from a sketch s with ground truth class y and target

class $y - N$. The target class is $y - N$ in this case because the generator wants the image $G(s)$ to be classified as a real image of the object depicted in sketch s and $y - N$ is the index of that class in the output vector. The second term in this loss is the $L1$ distance between the generated image $G(S)$ and the ground truth image x weighted by a hyper parameter λ. This $L1$ term encourages the generator to produce images that are close to the ground truth photo. The loss L_G is given by the equation:

$$L_G = -\mathbb{E}_{s,y\sim p_{data}(s,y)}[\log p_{model}(y - N|G(s), s, N < y \leq 2N)] + \lambda \mathcal{L}_{L1}(G) \tag{2}$$

3.3. Penalty Loss

The 2N cross entropy loss makes use of our 2N-dimensional output; however, it does not take into account much of the additional information provided by the 2N representation. For instance, it doesn't differentiate between a misclassification of the object category from a misclassification of realness. The penalty loss aims to make use of this additional information by weighting the cross entropy terms used in the 2N losses by constant penalty values, which vary depending on the type of misclassification. For a class prediction $\hat{y}$ with target class y our penalty function $pen(y, \hat{y})$ is as follows:

$$pen(y, \hat{y}) = \begin{cases} a; obj(y) = obj(\hat{y}), \text{is-fake}(y) \neq \text{is-fake}(\hat{y}) \\ b; obj(y) \neq obj(\hat{y}), \text{is-fake}(y) = \text{is-fake}(\hat{y}) \\ c; obj(y) \neq obj(\hat{y}), \text{is-fake}(y) \neq \text{is-fake}(\hat{y}) \end{cases}$$

where $obj()$ returns the type of object represented by the given class, is-fake() determines if the given class represents a fake image, and a, b, c are hyper parameters that can be chosen in cross validation. Using this penalty function we define our discriminator loss L_D to be:

$$\begin{aligned} L_D = -(&\mathbb{E}_{x,s,y\sim p_{data}(x,s,y)}[\log p_{model}(y|x, s, y \leq N)] \\ &\times pen(y, \hat{y}) \\ &+ \mathbb{E}_{s,y\sim p_{data}(s,y)}[\log p_{model}(y|G(s), s, N < y \leq 2N)] \\ &\times pen(y, \hat{y})) \end{aligned} \tag{3}$$

Our generator also weights the cross entropy term by the output of the penalty function and is given by the equation below. Note that we pass $y - N$ into the penalty function as the target class for the generated image $G(s)$ because the generator wants the image to be classified as a real image of the object depicted in sketch.

$$\begin{aligned} L_G = &\\ &- \mathbb{E}_{s,y\sim p_{data}(s,y)}[\log p_{model}(y - N|G(s), s, N < y \leq 2N)] \\ &\times pen(y - N, \hat{y}) + \lambda \mathcal{L}_{L1}(G) \end{aligned} \tag{4}$$

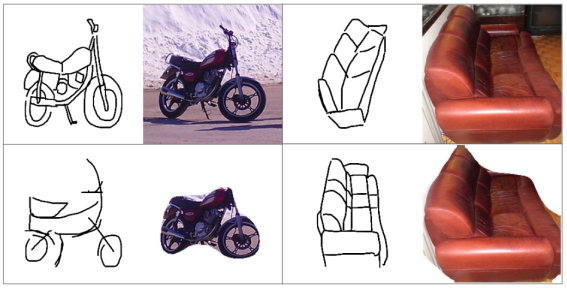

Figure 1: Examples of sketch-photo pairs. The bottom row displays examples of photos cropped using the segmentation mask.

4. Experiment

4.1. Dataset

We used the Sketchy Database [1], a large-scale collection of sketch-photo pairs created by Georgia Tech to perform image retrieval using deep learning. This database contains 12,500 images from a subset of 125 categories from Imagenet. The creators asked participants on Amazon Mechanical Turk to sketch the target object in the images, so that each image ended up with about 5 hand-drawn sketches for a total of 75,471 sketches in the final dataset. We eliminated 10,918 sketches that the creators had marked as ambiguous, erroneous, having an incorrect pose, or including environment details. Our final training size was 43,020 sketch-photo pairs.

4.2. Image Segmentation

During preliminary testing of our cGAN sketch-to-photo network, we noticed a consistent issue with our output images. As our image output population is comprised entirely of photographs, the images often have cluttered backgrounds. We surmised that often, our generator is learning to emulate the background instead of focusing on the requested object. In the class of $airplane$, this background emulation is not a problem, as here, most photo backgrounds are blue and uniform. The background becomes a greater issue in classes such as $eyeglasses$, where the image is cluttered with faces, hair, and other distracting elements. We hypothesized that by cropping our image set to only include the key object, we will see a much higher quality in the generated images.

In order to create a segmentation mask for our dataset, we adapted the findings proposed in "Fully Convolutional Networks for Semantic Segmentation" [4], using models created in "Deep Residual Learning for Instrument Segmentation in Robotic Surgery" [5]. This allowed us to utilize

[1] http://sketchy.eye.gatech.edu/

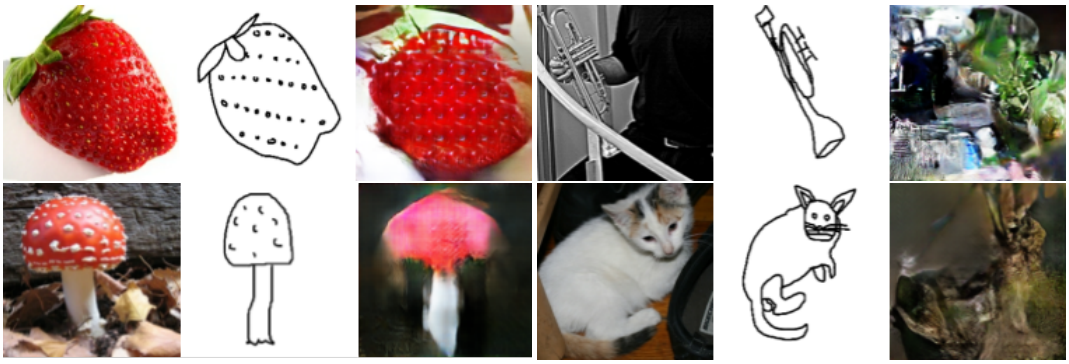

Figure 2: Images generated during training of penalty loss model. Successful generations on the left, unsuccessful generations on the right.

a model trained on the PASCAL VOC Image Segmentation Dataset. This model used 25 classes for segmentation, which limited the quantity of images we were able to effectively crop. By implementing the segmentation mask model on the Sketchy Database, we were able to produce a dataset with 15 object classes and over 9,000 coupled sketches and cropped images (see Figure 1). Due to time constraints, we only trained the baseline model with this segmented dataset.

4.3. Class Conditional Generator

In addition to our two proposed loss functions, we conducted an experiment in which we gave the target class as a conditional to the generator. We found that the crude sketches in our dataset often shared key features across classes. For example, sketches with a striped pattern often generated a "zebra-like" image with white and black stripes regardless of the rest of the sketch. By explicitly giving class information to the generator, we hoped to produce images that were more closely related to the class the sketch was based on.

We appended a one-hot encoding of the class to the vector produced at the end of encoding the input image. This modified vector was then decoded by the generator as in the above architecture to produce an image. Due to time constraints, we only trained the class conditional generator with the baseline loss functions.

4.4. Evaluation Methods

The state of the art for evaluating Generational Adversarial Networks is still being developed. We explore three different quantitative evaluation techniques to evaluate our models.

First, our implementation of this 2N output discriminator has the handy feature that it can double as a classifier. This is in contrast to a typical GAN discriminator that only outputs a realness classification. Utilizing this idea, we evaluated the accuracy of our trained discriminator as a classifier on real photos.

Next, we used the Inception score method introduced in [6], which applies a pre-trained Inception Network [7] to each generated image and computes the conditional class distribution. The distribution is expected to have low entropy for a single image, since we expect realistic images to be classified confidently by the network. The method also expects the marginal distribution across all generated images to be high, since the generated images ought to be varied from one another. These distributions are compared using KL-divergence, so that a higher Inception score indicates more realistic images. Previous work has found that the score for real images range from around 11.0 to 26.0, while scores reported for generated images range from around 8.0 to 9.0 [6, 8].

Finally, while the Inception score method is suitable for quantifying the realness of images generated by non-conditional GANs, it does not evaluate how competent a conditional generator is in generating class conditional images. To do so, we apply a pre-trained Inception Network to our generated images, and see whether the network is able to predict the image's conditional class. We calculate accuracy from the Top 1 and Top 5 classes predicted by the Inception Network.

4.5. Results

Subjectively observing our generated images, we saw that our models generated a variety of images. See Figure 2 for some examples of good and bad generated images. We found that categories that had little background noise (like $airplane$), showed the target object in a consistent shape or pose (like $mushroom$), or had consistent features such as color or texture (like $strawberry$) often generated better images than noisy, inconsistent categories (like musical instruments and animals). We also found that training on the segmented images successfully generated many images with similar shape to the target object. To see examples of

Model	Accuracy
2N Loss (50k steps)	26.98%
Penalty Loss (50k steps)	29.09%
Penalty Loss (134k steps)	10.26%

Table 1: Accuracy when classifying validation photos using the standalone discriminator.

Model	Mean	Std Dev
Ground Truth Photos	74.81	1.40
Baseline Model	5.26	0.16
Class Conditional Generator	4.25	0.07
2N Loss Model	6.11	0.11
Penalty Model	6.20	0.10
Segmented Photos	13.33	0.90
Trained on Segmented Photos	5.96	0.25

Table 2: Inception scores for various models.

images generated by all of the models, see Figure 3

To evaluate the discriminator as a standalone classifier, we classified 10,809 validation photos with the trained discriminators using the 2N loss and the penalty loss. The 2N loss discriminator was run for 50,000 iterations, and we used two penalty loss discriminators that trained for 50,000 iterations and 134,000 iterations respectively. See Table 1 for results. We were unable to compare these results to the baseline model because its loss function does not produce an output with class information. The results show that the accuracy decreases dramatically after training the penalty loss model for longer. We believe this is due to the real photos being misclassified as fake. Since we wish to test the discriminator as a classifier only on real photos, future work will classify images based on the first N elements of the output, which are the elements representing real class scores.

We computed the Inception score on 10,809 validation photos, the images generated from those sketch-photo pairs, 1,560 segmented validation photos, and the images generated from those sketch-segmented photo pairs using the five models explained above, each trained for 50,000 iterations. See Table 2 for results. While the Inception scores for our models are lower than scores previously reported by other papers due to the fewer training iterations, we see that the models using 2N loss and penalty loss slightly outperform the baseline model.

Results from classifying our generated images using an Inception network can be see in Table 3. While the low accuracy shows that there is still much room for improvement, since Imagenet has 1000 categories, our models are still being classified at a rate better than random. We see

Model	Top 1	Top 5
Ground Truth Photos	71.90%	79.04%
Baseline Model	0.83%	2.36%
Class Conditional Generator	1.05%	3.13%
2N Loss Model	0.48%	1.90%
Penalty Model	0.85%	2.44%
Segmented Photos	40.58%	60.51%
Trained on Segmented Photos	1.99%	4.42%

Table 3: Top 1 and Top 5 Accuracies for classifying generated images using Inception network.

higher accuracies for the class conditional generator model due to the generator explicitly receiving class information. We also see higher accuracies for the model trained on segmented photos, since that model excels in generating images that have shapes similar to the target object.

5. Conclusion

The results in this paper suggest that using a 2N class discriminator for cGANs has great promise, as these networks show results that are competitive with previously proposed methods. More work needs to be done to fully understand the potential of this approach. Additionally, this paper demonstrates the possibility of generating photographic images from an input of hand drawn sketches. We feel that both our understanding of the model and the quality of the generated photos would benefit from three clear next steps: implementing a conditional version of the N+1 class discriminator proposed in "Improved Techniques for Training GANs" to use as a baseline, training our models for longer (about 200 epochs), and learning penalty values for the penalty loss via cross validation. We hope that this work sparks interest in both using GANs to augment sketches and experimenting with a 2N class discriminator.

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 *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] 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.

[3] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1605.06211v1*, 2016.

[5] D. Pakhomov, V. Premachandran, M. Allan, M. Azizian, and N. Navab. Deep residual learning for instrument segmentation in robotic surgery. *arXiv preprint arXiv:1703.08580*, 2017.

[6] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[8] D. Warde-Farley and Y. Bengio. Improving generative adversarial networks with denoising feature matching. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[9] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

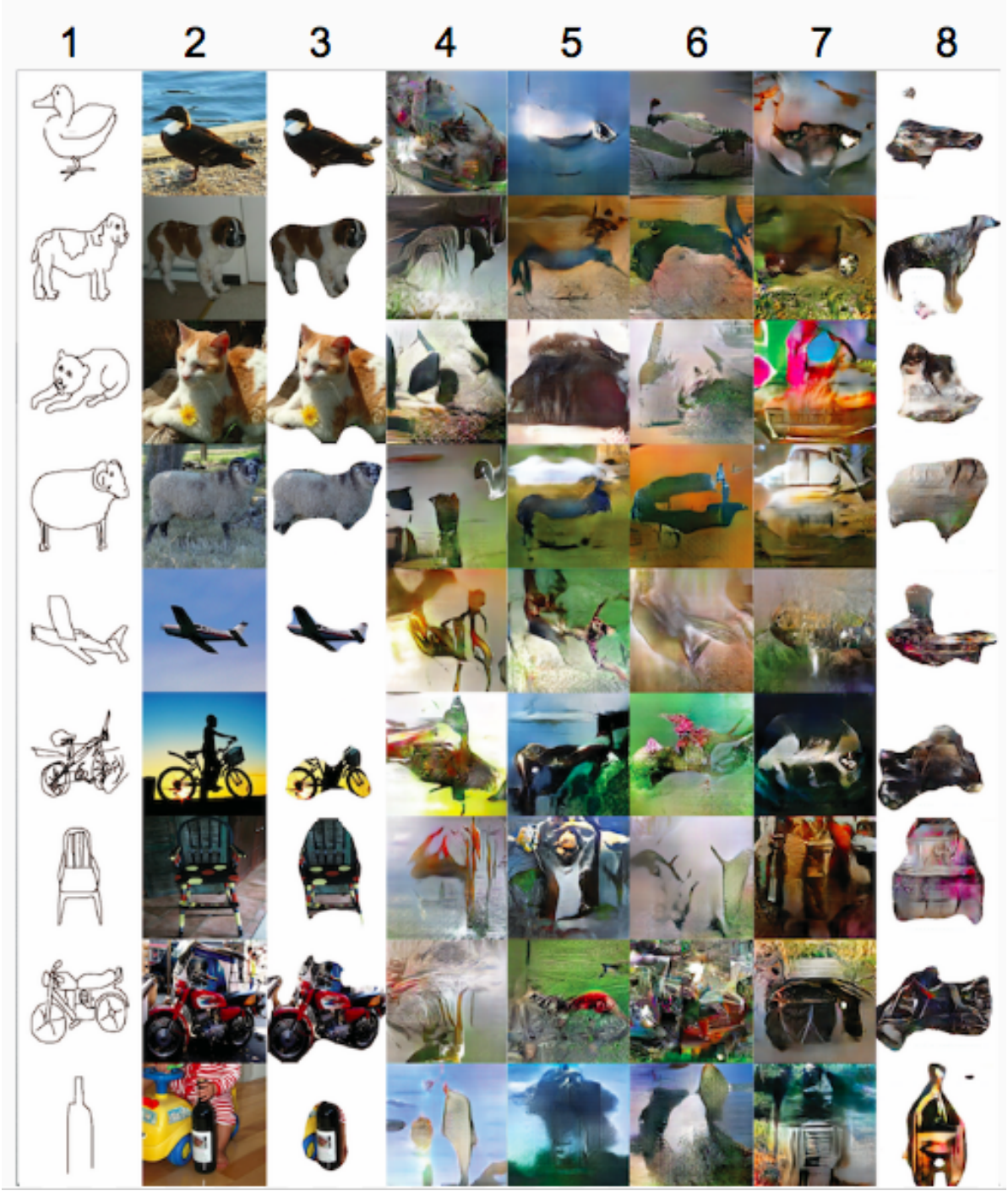

Figure 3: Examples of inputs and outputs of the various models. Each row corresponds to a sketch. The columns, from left to right, correspond to: 1. Input Sketches; 2. Target Photos; 3. Segmented Target Photos; 4. Baseline Model; 5. Class Conditional Generator; 6. 2N Loss Model; 7. Penalty Loss Model; 8. Trained on Segmented Photos.