Generative Vector Art

Universite de Montreal
IFT6756
Ivan Puhachov

Abstract

Vector graphics allows for editing without loss of quality. It stores images as a collection of primitives (lines, bezier curves, circles, etc.) with corresponding properties (size, width, control points).
Typical generative models are trained on raster images, while vector graphics space is non-euclidean. Several approaches were tried, until finally a differentiable rasterizer proposed in [6] made it possible to do a direct backpropagation to svg parameters.

In this project we apply differentiable rasterizer from [6] to experiment with generative art on different datasets, GAN architectures, and create adversarial examples using painting.

Related work

Most generative adversarial networks are generating raster images. DoodlerGAN [3] aimed to replicate human drawings by generating raster body parts.
SketchRNN [5] modeled drawing as a decision making sequential process and trained seq2seq RNN VAE to generate quick sketches. SVG VAE [8] applies the same approach to model fonts as a sequence of 4 commands, SketchBERT [7] moved to a next level by training a giant BERT model on the same data. DeepSVG [2] uses transformers to generate SVG control commands for a vider family of svg primitives.
Differentiable rasterizer proposed in [6] allows gradient flow from rendered image to svg parameteres, thus we can optimize svg primitives directly from raster signal. They also showed several application of generative models. Im2Vec [10] uses this rasterizer to generate vector graphics from raster examples.

GAN architectures

Apart from playing with "vanilla" GAN (with and without convolutions) we tried the following approaches:
WGAN [1] makes a critic K-Lipschitz by clipping its gradients to stabilize training

$$\omega \leftarrow \text{clip}(\omega, -0.01, 0.01)$$

WGAN-GP [4] forces K-Lipschitz by adding a gradient penalty to the loss

$$\hat{x} \leftarrow \epsilon x_{real} + (1-\epsilon)x_{fake}$$

$$\text{loss} \leftarrow D(x_{fake}) - D(x_{real}) + \lambda(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2$$

SNGAN [9] Force 1-Lipschitz continuity by spectral normalization of critic's layers

$$\bar{W}_{SN}(W) = \frac{W}{\sigma(W)}$$

LOGAN [11] More gradient flow to generator by making a latent gradient step $z = z + \Delta z$ before passing image to discriminator

$$g = \frac{\partial D(G(z))}{\partial z}$$

$$\Delta z = \frac{\alpha}{\beta + \|g\|^2} \cdot g$$

Results LOGAN suffered from mode collapse and instabilities, and we haven't managed to make it train.

Model	**IS**	**FID**
WGAN	1.935	94.14
WGAN-GP	1.856	101.7
SN WGAN-GP	1.919	84.54
LOGAN NGD	1.11	208

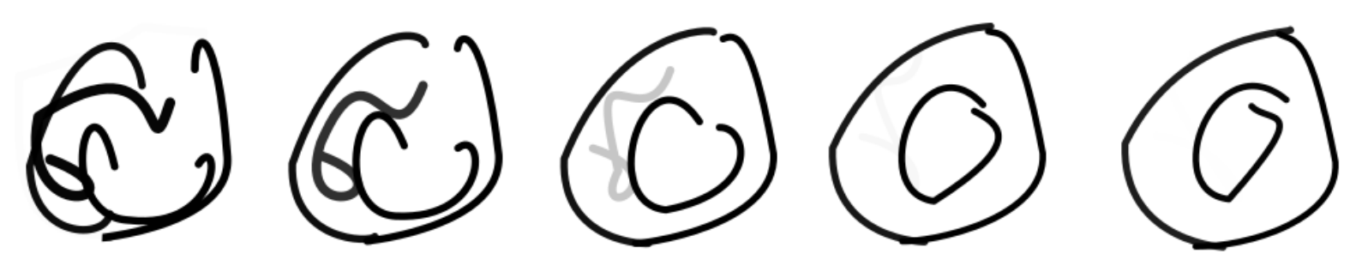
While Inception Score and Frechet Inception Distance are good for automatic evaluation of generative models for realistic images, we found them less informative for grayscale sketches.

GAN details

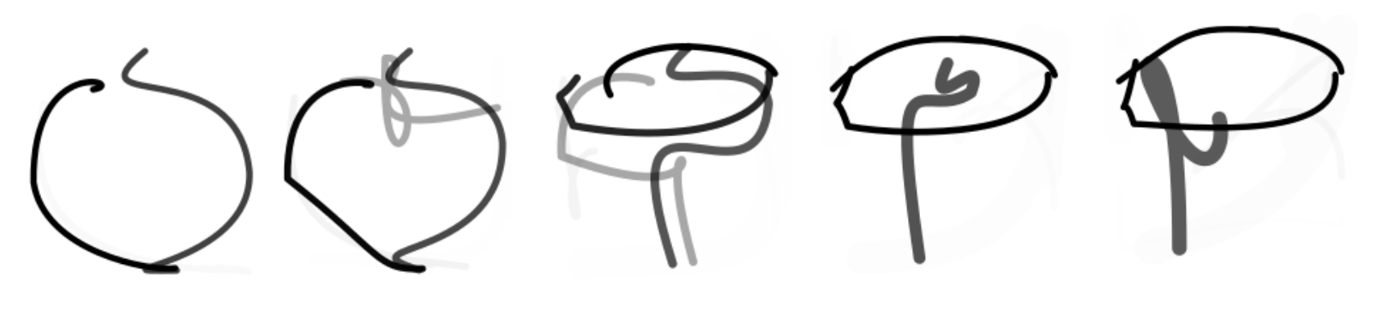
Training time Rasterization is expensive, (i.e. iterative root solving for high-degree polynomials), and it affects training.
Two Time-Scale Update Rule Helped stabilize training. In simple words, the idea is to put smaller learning rate for the generator.
Inception Score To evaluate generation results we used two metrics. Inception score takes pre-trained inception model and measures the distribution of predicted labels over generated samples.
Frechet Inception Distance Evaluate feature distribution of a pre-trained inception model over real data and generated samples.

$$\|\mu_{real} - \mu_{fake}\|_2^2 + \text{Trace}(\Sigma_{real} + \Sigma_{fake} - 2(\Sigma_1\Sigma_2)^{0.5})$$

Complex dataset We tried training VectorGAN on a dataset of complex images. It takes a lot more time to converge to meaningful drawings.
Draw with more primitives Discrete decisions (such as number of primitives and their types) are still non-differentiable operation. Our model works with bezier curves (x16) and circles (x5), as other primitives are rare in the dataset.
Interpolation Having 2 latent vectors, we can interpolate between images. Note that although model was trained on 64x64 images, we can generate image of any resolution, thanks to the vector graphics.

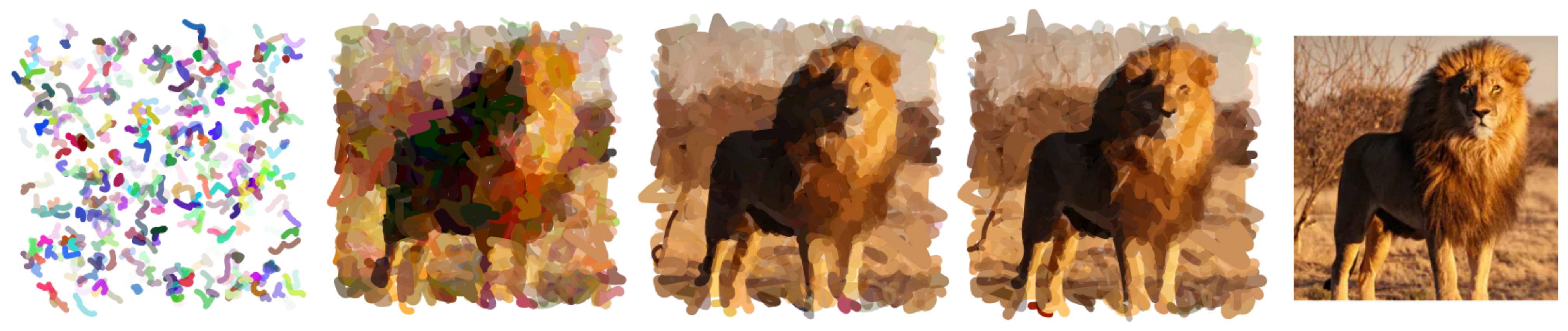


Meaningful directions This is an area for future experiments. Now, as the generator has direct access to positions and control parameters, disentangling latent space directions seems easier. Is it really?

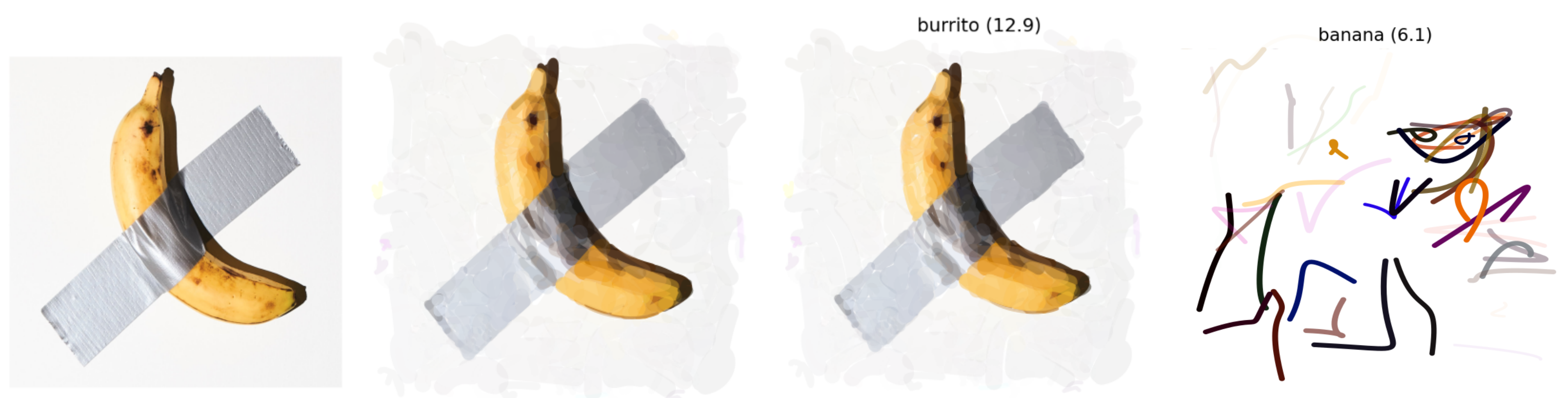


Generative Art

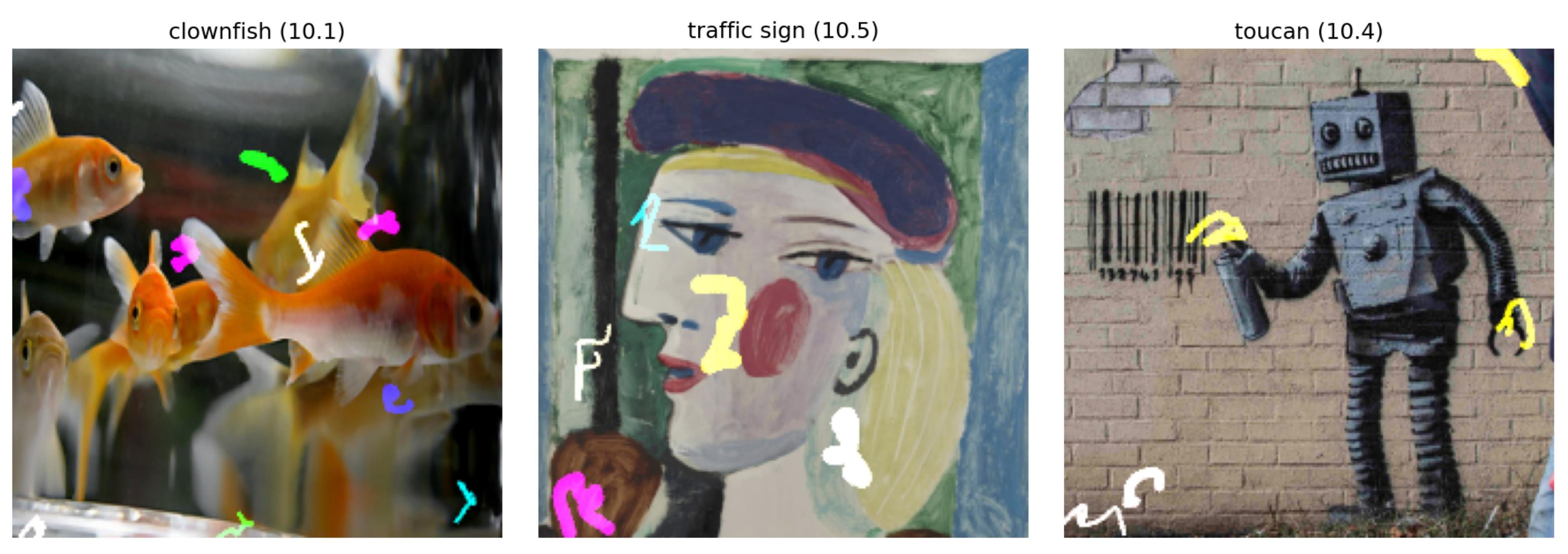
Painterly rendering Fit svg parameters to match with raster image. Discrete decisions (number of strokes, types) is still fixed by a human.



Adversarial creator Update parameters of svg to fool InceptionV3 into wrong classification. For example, fast gradient sign method or just plain gradient descent. We can also fix some parameters, and update only colors, or control points, or widths.
Initial idea was to have small number of strokes and, starting from random, move it towards inception models gradients. However, it produces adversarial example with little artistic meaning, consider picture on the right.



Adversarial vandal Draw on top of a raster picture to fool InceptionV3. Update svg parameters to maximize target class score.



[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.
[2] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020.
[3] Songwei Ge, Vedanuj Goswami, C. Lawrence Zitnick, and Devi Parikh. Creative sketch generation. 2020.
[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.
[5] David Ha and Douglas Eck. A neural representation of sketch drawings. In *ICLR*, 2018.
[6] Tzu-Mao Li, Michal Lukac, Michael Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph.*, 39(6), November 2020.
[7] Hangyu Lin, Yanwei Fu, Yu-Gang Jiang, and X. Xue. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6757–6766, 2020.
[8] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics, 2019.
[9] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.
[10] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. Im2vec: Synthesizing vector graphics without vector supervision, 2021.
[11] Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. Logan: Latent optimisation for generative adversarial networks, 2020.

Ivan Puhachov
ivan.puhachov@umontreal.ca
Universite de Montreal