A. Optimal D for Conditional Generation

It turns out that in the case of conditional generation (*i.e.*, p_d is a Dirac δ-function), we can derive an explicit form of the optimal (non-parametric) discriminator D. Indeed, (1) corresponds to the dual representation of the Wasserstein-1 metric (Villani, 2008). The primal form of that metric is defined as

$$W_1(p_g, p_d) = \inf_{\gamma \in \Gamma(p_g, p_d)} \int \|\mathbf{x} - \mathbf{y}\|_2 \, d\gamma(\mathbf{x}, \mathbf{y}) \,, \tag{8}$$

where $\Gamma(p_g, p_d)$ is a set of all couplings between p_g and p_d. Taking into account that the data distribution is a point mass, we can simplify (8):

$$W_1(p_g, p_d) = \mathbb{E}_{\mathbf{x} \sim p_g} \|\mathbf{x} - \mathbf{x}_{\text{target}}\|_2 \,. \tag{9}$$

The expression above gives the optimal value for $\mathcal{L}_D$ in (1). Therefore $D(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_{\text{target}}\|_2$ is a solution of (1):

$$\begin{aligned} \mathcal{L}_D(\|\mathbf{x} - \mathbf{x}_{\text{target}}\|_2) = &- \|\mathbf{x}_{\text{target}} - \mathbf{x}_{\text{target}}\|_2 \\ &+ \mathbb{E}_{\mathbf{x} \sim p_g} \|\mathbf{x} - \mathbf{x}_{\text{target}}\|_2 \\ &+ 0 \,, \end{aligned} \tag{10}$$

where the last term (R) is zero since the Euclidean distance belongs to the set of 1-Lipschitz functions.

This result suggests that for inverse graphics, in (4), one may use a fixed image distance (like the Euclidean distance ℓ^2) instead of a parametric function optimized via the WGAN objective. Note, however, that ℓ^2 is not a unique solution to (1). Consider, for example, the case where the model distribution is also a Dirac delta centered at $\mathbf{x}_\text{g}$. The Wasserstein distance is equal to $d = \|\mathbf{x}_\text{g} - \mathbf{x}_{\text{target}}\|_2$. In order to achieve that value in (1), we could take any D such that $\forall\, \alpha \in [0, 1]$

$$D\left(\alpha\, \mathbf{x}_\text{g} + (1 - \alpha)\, \mathbf{x}_{\text{target}}\right) = \alpha \cdot d \,, \tag{11}$$

and $\text{Lip}(D) \leq 1$. One example of such function would be a hyperplane H containing the segment (11). Let us now consider a set of points

$$V = \left\{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}_\text{g}\|_2 < \varepsilon, \|\mathbf{x} - \mathbf{x}_{\text{target}}\|_2 = d\right\} \tag{12}$$

in an ε-vicinity of $\mathbf{x}_\text{g}$. By definition, $\|\mathbf{x} - \mathbf{x}_{\text{target}}\|_2$ is constant for any $\mathbf{x} \in V$. That means that ℓ^2 expresses no preference over points that are equidistant from $\mathbf{x}_{\text{target}}$ even though some of them may be semantically closer to $\mathbf{x}_{\text{target}}$. This property may significantly slow down learning if we are relying on ℓ^2 (or similar distance) as our training signal. Functions like H, on the other hand, have non-zero slope in V and therefore can potentially shift the search towards more promising subspaces.

One other reason why discriminator training is different from using a fixed image distance is that in practice, we do not optimize the exact dual formulation of the Wasserstein distance and, on top of that, use stochastic gradient descent methods which we do not run until convergence. A toy example illustrating that difference is presented in Figure 11.

B. Network Architectures

The policy network (shown in Figure 12) takes the observation (*i.e.*, the current state of the canvas C_t) and conditions it on a tuple corresponding to the last performed action a_t. The resulting features are then downsampled to a lower-dimensional spatial resolution by means of strided convolutions and passed through a stack of ResNet blocks (He et al., 2016) followed by a fully-connected layer. This yields an embedding which we feed into an LSTM (Hochreiter & Schmidhuber, 1997). The LSTM produces a hidden vector z_0 serving as a seed for the action sampling procedure described below.

In order to obtain a_{t+1}, we employ an *autoregressive decoder* depicted in Figure 13. Each component a_{t+1}^i is sampled from a categorical distribution whose parameters are computed as a function of z_i. We use two kinds of functions depending on whether a_{t+1}^i corresponds to a scalar (*e.g.*, brush size) or to a spatial location (*e.g.*, a control point of a Bézier curve). In the scalar case, z_i is transformed by a fully-connected layer, otherwise we process it using several ResNet blocks followed by a series of transpose convolutions and a final convolution. After a_{t+1}^i is sampled, we obtain an updated hidden vector z_{i+1} by embedding a_{t+1}^i into a 16-dimensional code and combining it with z_i. The procedure is repeated until the entire action tuple has been generated.

For the discriminator network, we use a conventional architecture similar to DCGAN (Radford et al., 2015).

C. Training Details

Following standard practice in the GAN literature, we optimize the discriminator objective using Adam (Kingma & Ba, 2014) with a learning rate of 10^{-4} and β_1 set to 0.5. For generator training, we employ population-based exploration of hyperparameters (PBT) (Jaderberg et al., 2017) to find values for the entropy loss coefficient and learning rate of the policy learner. A population contains 12 training instances with each instance running 64 CPU actor jobs and 2 GPU jobs (1 for the policy learner and 1 for the discriminator learner). We assume that discriminator scores are compatible across different instances and use them as a measure of fitness in the exploitation phase of PBT.

The batch size is set to 64 on both the policy learner and discriminator learner. The generated data is sampled uniformly from a replay buffer with a capacity of 20 batches.

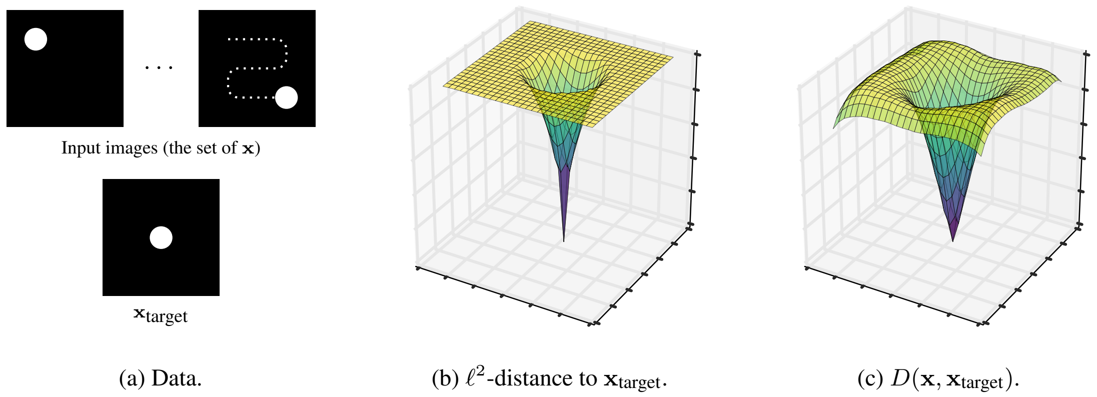

Figure 11. A toy experiment illustrating the **difference between ℓ^2 and discriminator** training in practice. **(a)** We collect a dataset of images with a single solid circle in all possible locations (top) and pick one of them as a target image (bottom). **(b)** The ℓ_2-distance (in the pixel space) from the input images to the target as a function of the circle location; the surface is flat around the borders since the circles do not overlap. **(c)** We train a discriminative model D that takes a pair of images and tells whether they match or not; just like ℓ^2, the resulting function has a pit centered at the target location, but unlike (b), the surface now has better behaved gradients.

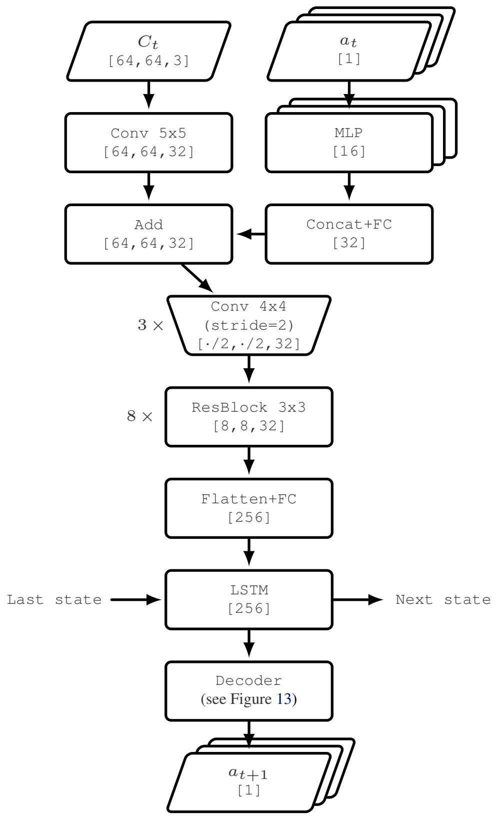

Figure 12. The architecture of the **policy network** for a single step. `FC` refers to a fully-connected layer, `MLP` is a multilayer perceptron, `Conv` is a convolutional layer and `ResBlock` is a residual block. We give the dimensions of the output tensors in the square brackets. ReLU activations between the layers have been omitted for brevity.

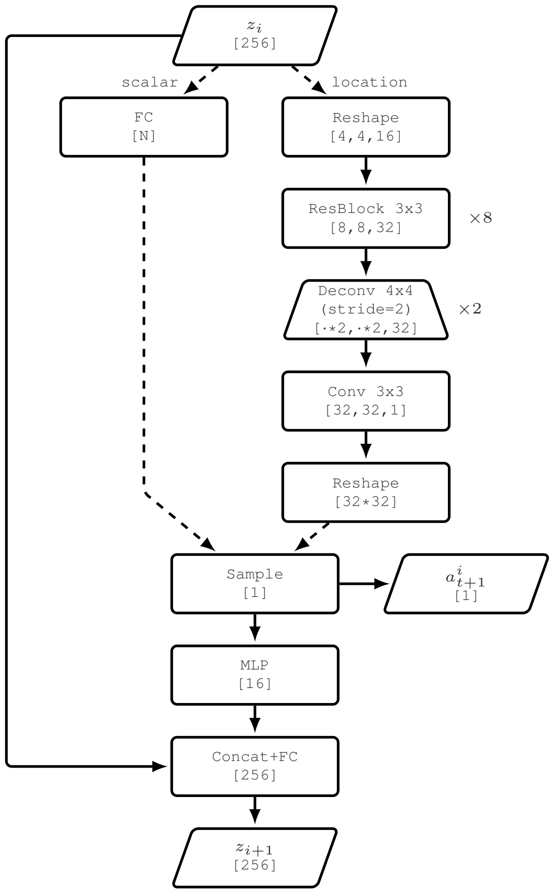

Figure 13. The architecture of the **autoregressive decoder** for sampling an element a^i_{t+1} of the action tuple. The initial hidden vector z_0 is provided by an upstream LSTM. Depending on the type of the subaction to be sampled, we use either the `scalar` or the `location` branch of the diagram.