# Supplementary Material : Adversarial Learning of a Sampler Based on an Unnormalized Distribution

## A Proof of the Entropy Bound in Lemma 1

Consider random variables $(x, \epsilon)$ under the joint distribution $q_\theta(x, \epsilon) = q(\epsilon)q_\theta(x|\epsilon)$, where $q_\epsilon(x|\epsilon) = \delta(x - h_\theta(\epsilon))$. The mutual information between $x$ and $\epsilon$ satisfies $I(x; \epsilon) = H(x) - H(x|\epsilon) = H(\epsilon) - H(\epsilon|x)$. Since $q_\theta(x|\epsilon)$ is a deterministic function of $\epsilon$, $H(x|\epsilon) = 0$. We therefore have $H(x) = H(\epsilon) - H(\epsilon|x)$, where $H(\epsilon) = -\int q(\epsilon) \log q(\epsilon) \mathrm{d}\epsilon$ is a constant wrt $\theta$. For general distribution $t_\xi(\epsilon|x)$,

$$H(\epsilon|x) = -\mathbb{E}_{p_\theta(x,\epsilon)} \log p_\theta(\epsilon|x) \tag{12}$$

$$= -\mathbb{E}_{q_\theta(x,\epsilon)} \log t_\xi(\epsilon|x) - \mathbb{E}_{q_\theta(x)} \mathrm{KL}(q_\theta(\epsilon|x) \| t_\xi(\epsilon|x))$$

$$\leq -\mathbb{E}_{q_\theta(x,\epsilon)} \log t_\xi(\epsilon|x) \tag{13}$$

We consequently have

$$H(x) = -\mathbb{E}_{q_\epsilon(x)} \log q_\theta(x) \mathrm{d}x$$

$$= H(\epsilon) - H(\epsilon|x) \geq H(\epsilon) + \mathbb{E}_{p_\theta(x,\epsilon)} \log t_\xi(\epsilon|x). \tag{14}$$

Therefore, entropy is lower bounded by the log likelihood or negative cycle-consistency loss; minimizing the cycle-consistency loss maximizes the entropy or mutual information. □

## B Experiments

### B.1 Sampling from 8-GMM

Two methods are presented for estimating the likelihood ratio: (i) $\sigma$-ALL for the discriminator in the standard GAN *i.e.,* Eq (4); (ii) $f$-ALL for a variational characterization of $f$-measures in [Nguyen et al., 2010a].

In Figure 8, we plot the distribution of inception score (ICP) values [Li et al., 2017a]. Similar conclusions as in the case of the symmetric KL divergence metric the can be drawn: (1) The likelihood ratio impelmentation improve the original GAN, and (2) the entropy regularizer improve the all GAN variants. Note that because ICP favors the samples closer to the mean of each mode and SN-GAN generate samples that concentrate only around the mode's centroid, SN-GAN show slightly better ICP than its entroy-regularized version. We argue that the entropy regualizer help gernerate diverse samples, the lower value of ICP is just due to the limitation of the metric.

The learning curves of the inception score and symmetric KL divergence values are plot over iterations in Figure 9 (a) and (b), respectively. The family of GAN variants with entropy term dominate the performance, compared with those without the entropy term. We conclude that the entropy regularizer can significantly improve the convergence speed and the final performance.

**Architectures and Hyper-parameters** For the 8-GMM and MNIST datasets, the network architectures are specified in Table 2, and hyper-parameters are detailed in Table 3. The inference network is used to construct the cycle-consistency loss to bound the entropy.

Table 2: The convention for the architecture "X–H–H–Y": X is the input size, Y is the output size, and H is the hidden size. "ReLU" is used for all hidden layer, and the activation of the output layer is linear, except the generator on MNIST is the sigmoid

| | 8-GMM | MNIST |
|---|---|---|
| Networks | Size | Size |
| Generator | 2–128–128–2 | 32–256–256–784 |
| Discriminator | 2–128–128–1 | 784–256–256–1 |
| Auxiliary | 2–128–128–2 | 784–256–256–32 |

Table 3: The hyper-parameters of experiments. Adam optimizer is used.

| Hyper-parameters | 8GMM | MNIST |
|---|---|---|
| Learning rate | $2\times10^{-4}$ | $1\times10^{-3}$ |
| Batch Size | 1024 | 64 |
| #Updates | 50k Iterations | 60 Epoches |

We further study three real-world datasets of increasing diversity and size: MNIST, CIFAR10 [Krizhevsky et al., 2012] and CelebA [Liu et al., 2015]. For each dataset, we start with a standard GAN model: two-layer fully connected (FC) networks on MNIST, as well as DCGAN [Radford et al., 2016] on CIFAR and CelebA. We then add the entropy regularizer. On MNIST, we

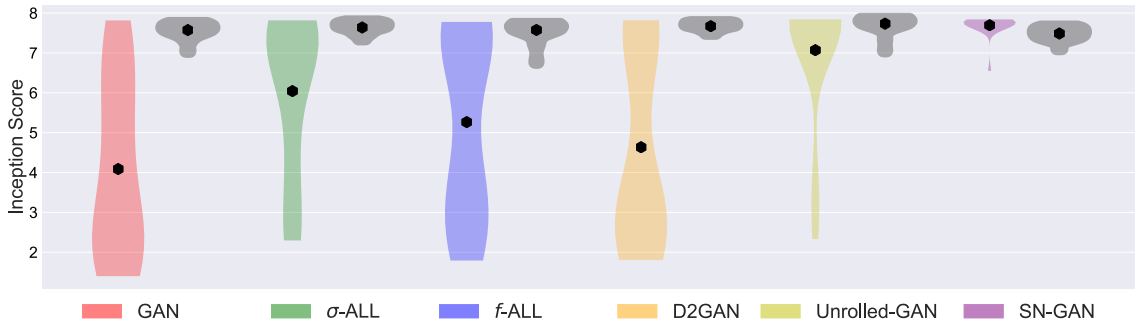
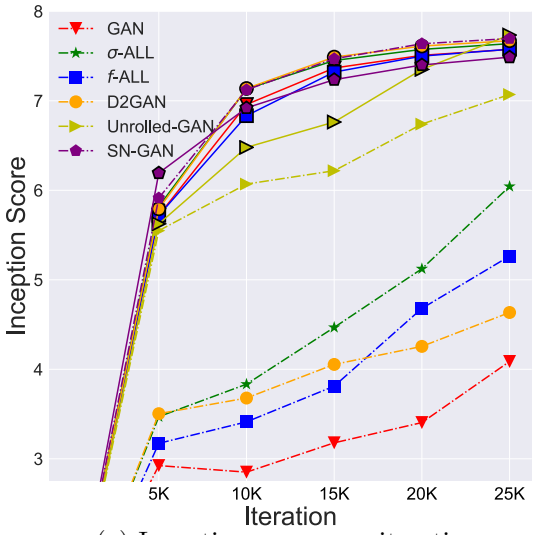
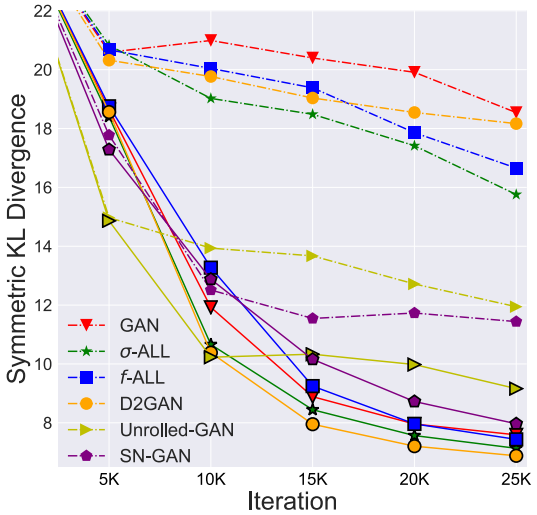



Figure 8: Comparison of inception score on different GAN variants. The GAN variants and their corresponding entropy-regularized variants are visualized in the same color, with the latter shaded slightly. The balck dots indicate the means of the distributions.




(a) Inception score over iterations.




(b) Symmetric KL over iterations.

Figure 9: Learning curves of different GAN variants. The standard GAN variants are visualized as dashed lines, while their corresponding entropy-regularized variants are visualized as the solid lines in the same color.

repeat the experiments 5 times, and the mean ICP is shown. On CIFAR and CelebA, the performance is also quantified via the recently proposed Fréchet Inception Distance (FID) [Heusel et al., 2017], which approximates the Wasserstein-2 distance of generated samples and true samples. The best ICP and FID for each algorithm are reported in Table 4. The entropy variants consistently show better performance than the original counterparts.

Table 4: Performance of entropy regularization. Results marked with [⋆] and [⋄] are from [Nguyen et al., 2017] and [Heusel et al., 2017], respectively.

| | ICP ↑ | | FID ↓ | |
|---|---|---|---|---|
| Dataset | Standard | $E_{cc}$ | Standard | $E_{cc}$ |
| MNIST | 7.24 | **8.08** | - | - |
| CIFAR | 6.40⋆ | **6.86** | 36.90⋄ | **36.70** |
| CelebA | - | - | 12.50⋄ | **11.88** |

### B.2 Constrained Domains

The two functions are: (1) $u_1(x) = \max((1-(x/2+0.5))(x/2+0.5)^3, 0)$, and (2) $u_2(x) = \max((1-(x/2+0.5))^{0.5}(x/2+0.5)^5 + (1-(x/2+0.5))^5(x/2+0.5)^{0.5}, 0)$. The network architectures used for constrained domains are reported in Table 5. The batch size is 512, learning rate is $1 \times 10^{-4}$. The total training iterations $T$ = 20k, and we start to decay $\beta$ after $T_0$ = 10k iterations.

Table 5: The convention for the architecture "X–H–H–Y": X is the input size, Y is the output size, and H is the hidden size. "ReLU" is used for all hidden layer, and the activation of the output layer is "Tanh".

| Networks | Size |
|---|---|
| Generator | 2–128–128–1 |
| Discriminator | 2–128–128–1 |
| Auxiliary | 1–128–128–2 |

**Algorithm 1** Adversarial Soft Q-learning

**Require:** Create replay memory $\mathcal{D} = \emptyset$; Initialize network parameters $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}$; Assign target parameters: $\overline{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$, $\overline{\boldsymbol{\psi}} \leftarrow \boldsymbol{\psi}$.

**for** each epoch **do**
  **for** each $t$ **do**
    % Collect expereince
    Sample an action for $\boldsymbol{s}_t$ using $g^{\boldsymbol{\theta}}$: $\boldsymbol{a}_t \leftarrow g^{\boldsymbol{\theta}}(\boldsymbol{\xi}; \boldsymbol{s}_t)$, where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    Sample next state and reward from the environment: $\boldsymbol{s}_{t+1} \sim P_{\boldsymbol{s}}$ and $r_t \sim P_r$
    Save the new experience in the replay memory: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}\}$
    % Sample a minibatch from the replay memory
    $\{(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}, r_t^{(i)}, \boldsymbol{s}_{t+1}^{(i)})\}_{i=0}^{n} \sim \mathcal{D}$.
    % Update Q value network
    Sample $\{\boldsymbol{a}^{(i,j)}\}_{j=0}^{M} \sim q_{\boldsymbol{a}'}$ for each $\boldsymbol{s}_{t+1}^{(i)}$.
    Compute the soft Q-values $u(\boldsymbol{a}, \boldsymbol{s})$ as the target unnormalized density form.
    Compute gradient of Q-network and update $\boldsymbol{\psi}$
    % Update policy network via RAS
    Sample actions for each $\boldsymbol{s}_t^{(i)}$ from the stochastic policy via
$$\boldsymbol{a}_t^{(i,j)} = f^{\boldsymbol{\phi}}(\boldsymbol{\xi}^{(i,j)}, \boldsymbol{s}_t^{(i)}), \text{ where } \{\boldsymbol{\xi}^{(i,j)}\}_{j=0}^{M} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$
    Sample actions for each $\boldsymbol{s}_t^{(i)}$ from a Beta (or Gaussian) reference policy $\{\boldsymbol{a}_r^{(i,j)}\}_{j=0}^{M} \sim p_r(\boldsymbol{a}|\boldsymbol{s}_t^{(i)})$
    Compute gradient of discriminator in (8) and update $\boldsymbol{\phi}$
    Compute gradient of policy network in (9), and update $\boldsymbol{\theta}$
  **end for**
  **if** epoch *mod* update_interval $= 0$ **then**
    Update target parameters: $\overline{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$, $\overline{\boldsymbol{\psi}} \leftarrow \boldsymbol{\psi}$
  **end if**
**end for**

### B.3 Soft Q-learning

We show the detailed setting of environments in Soft Q-Learning in Table 6. The network architectures are specified in Table 7, and hyper-parameters are detailed in Table 8. We only add the entropy regularization at the beginning to stabilize training, and then quickly decay $\beta$ to 0. The total training epoch is 200, and we start to decay $\beta$ after 10 epochs, and set it 0 after 50 epochs. This is because we observed that the entropy regularization did not help in the end, and removing it could accelerate training.

Table 6: Hyper-parameters in SQL.

| Environment | Action Spcae | Reward Scale | Replay Pool Size |
|---|---|---|---|
| Swimmer (rllab) | 2 | 100 | $10^6$ |
| Hopper-v1 | 3 | 1 | $10^6$ |
| HalfCheetah-v1 | 6 | 1 | $10^7$ |
| Walker2d-v1 | 6 | 3 | $10^6$ |
| Ant-v1 | 8 | 10 | $10^6$ |
| Humanoid (rllab) | 21 | 100 | $10^6$ |

Table 7: The convention for the architecture "X–H–H–Y": X is the input size, Y is the output size, and H is the hidden size. "ReLU" is used for all hidden layer, and the activation of the output layer is "Tanh" for the policy network and linear for the others. $\mathcal{S}$ represents the state, $\mathcal{A}$ represents the action. $\mathcal{N}$ is the gaussian noise. The dimension of the noise is the same as the action space. The parameters settings of SVGD version and ours version are the same.

| Networks | Size |
|---|---|
| Policy-Network | $\lvert\mathcal{S}+\mathcal{N}\rvert$–128–128–$\lvert\mathcal{A}\rvert$ |
| Q-Network | $\lvert\mathcal{S}+\mathcal{A}\rvert$–128–128–1 |
| Inverse Mapping | $\lvert\mathcal{A}\rvert$–128–128–$\lvert\mathcal{S}+\mathcal{N}\rvert$ |
| Discriminator | $\lvert\mathcal{A}+\mathcal{S}\rvert$–128–128–128–1 |

Table 8: The hyper-parameters of experiments.

| Hyper-parameters | Values |
|---|---|
| Learning rate of Policy | $3\times10^{-4}$ |
| Learning rate of Q-network | $3\times10^{-4}$ |
| Batch Size | 128 |
| #Particle in SVGD | 32 |