Summary of A Few Recent Papers about Discrete Generative models

Presenter: Ji Gao

Department of Computer Science, University of Virginia
https://qdata.github.io/deep2Read/

Outline

- SeqGAN
- BGAN: Boundary Seeking Generative Adversarial Networks
- MaskGAN
- BEGAN

SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

- Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu
- Shanghai Jiaotong University

SeqGAN: Policy gradient + MC search

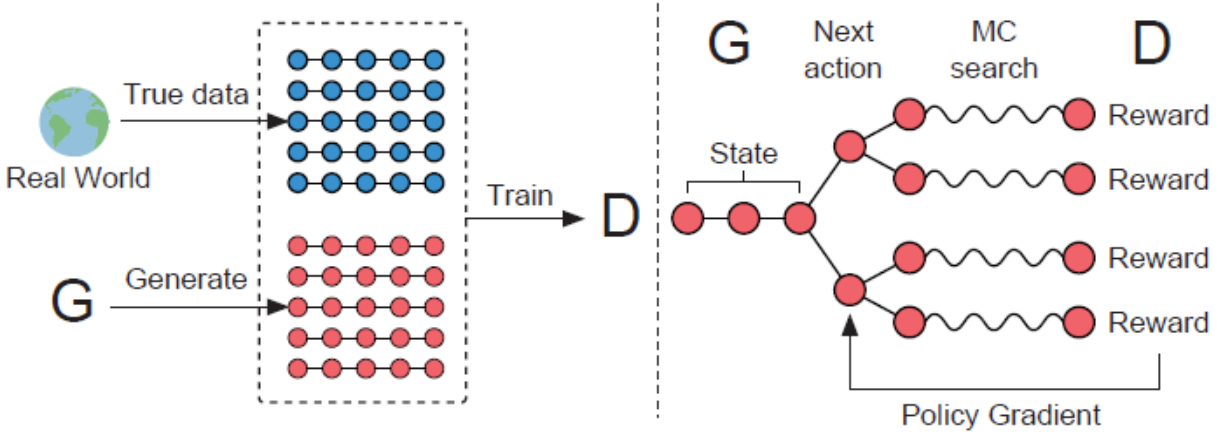


Figure 1: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

SeqGAN: Notations

- Generator G_θ, Discriminator D_ϕ
- Goal: Generate token sequence $Y_{1:T} = (y_1 .. y_T), y_t \in \mathcal{Y}$
- In time step t, state s is preceding tokens $y_1 .. y_{t-1}$, and action a is the next token y_t
- Policy: $G_\theta(y_t | Y_{1:t-1})$ is a stochastic policy over all possible tokens
- GAN loss: $L(\theta, \phi) = -E_{Y \sim p_{data}}[\log D(Y)] - E_{Y \sim p_g}[\log(1 - D(Y))]$
- For the generator in training, loss is $L_g(\theta) = -E_{Y \sim p_g}[\log D(Y)]$

Problem: No way to train the generator

- What we want: $\nabla_\theta \log D(Y) = \frac{1}{D(Y)} \frac{\partial D(Y)}{\partial Y} \nabla_\theta Y$
- However, for discrete Y, there's no $\nabla_\theta Y$
- Unable to train the generator directly

In the RL view

Suppose $Q^{G_\theta}(s, a)$ is the value function, that is, the expected accumulative reward start from state s taking policy G_θ.

$$J(\theta) = E[R_T | s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

We have:

$$Q_{D_\phi}^{G_\theta}(Y_{1:T-1}, y_T) = D_\phi(Y_{1:T})$$

The whole process only get reward at the end of the process. $Q^{G_\theta}(s, a)$

Monte Carlo Tree Search

Use N-time Monte Carlo Tree Search:

$$\{Y_{1:T}^{1}, \ldots, Y_{1:T}^{N}\} = MC^{G_\theta}(Y_{1:t}\,; N)$$

We have:

$$Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^{n}), Y_{1:T}^{n} \in MC^{G_\beta}(Y_{1:t}; N) \text{ for } t < T \\ D_\phi(Y_{1:t}) \text{ for } t = T \end{cases}$$

Policy Gradient

- G – Generator(probabi
- Q – Value function
- V – State Value

$$
\begin{aligned}
&\nabla_\theta J(\theta)\\
&= \nabla_\theta V^{G_\theta}(s_0) = \nabla_\theta[\sum_{y_1\in\mathcal{Y}} G_\theta(y_1|s_0)\cdot Q^{G_\theta}(s_0,y_1)]\\
&= \sum_{y_1\in\mathcal{Y}} [\nabla_\theta G_\theta(y_1|s_0)\cdot Q^{G_\theta}(s_0,y_1) + G_\theta(y_1|s_0)\cdot \nabla_\theta Q^{G_\theta}(s_0,y_1)]\\
&= \sum_{y_1\in\mathcal{Y}} [\nabla_\theta G_\theta(y_1|s_0)\cdot Q^{G_\theta}(s_0,y_1) + G_\theta(y_1|s_0)\cdot \nabla_\theta V^{G_\theta}(Y_{1:1})]\\
&= \sum_{y_1\in\mathcal{Y}} \nabla_\theta G_\theta(y_1|s_0)\cdot Q^{G_\theta}(s_0,y_1) + \sum_{y_1\in\mathcal{Y}} G_\theta(y_1|s_0)\nabla_\theta[\sum_{y_2\in\mathcal{Y}} G_\theta(y_2|Y_{1:1})Q^{G_\theta}(Y_{1:1},y_2)]\\
&= \sum_{y_1\in\mathcal{Y}} \nabla_\theta G_\theta(y_1|s_0)\cdot Q^{G_\theta}(s_0,y_1) + \sum_{y_1\in\mathcal{Y}} G_\theta(y_1|s_0) \sum_{y_2\in\mathcal{Y}} [\nabla_\theta G_\theta(y_2|Y_{1:1})\cdot Q^{G_\theta}(Y_{1:1},y_2)\\
&\qquad + G_\theta(y_2|Y_{1:1})\nabla_\theta Q^{G_\theta}(Y_{1:1},y_2)]\\
&= \sum_{y_1\in\mathcal{Y}} \nabla_\theta G_\theta(y_1|s_0)\cdot Q^{G_\theta}(s_0,y_1) + \sum_{Y_{1:1}} P(Y_{1:1}|s_0;G_\theta) \sum_{y_2\in\mathcal{Y}} \nabla_\theta G_\theta(y_2|Y_{1:1})\cdot Q^{G_\theta}(Y_{1:1},y_2)\\
&\qquad + \sum_{Y_{1:2}} P(Y_{1:2}|s_0;G_\theta)\nabla_\theta V^{G_\theta}(Y_{1:2})\\
&= \sum_{t=1}^{T} \sum_{Y_{1:t-1}} P(Y_{1:t-1}|s_0;G_\theta) \sum_{y_t\in\mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1})\cdot Q^{G_\theta}(Y_{1:t-1},y_t)\\
&= \sum_{t=1}^{T} \mathbb{E}_{Y_{1:t-1}\sim G_\theta}[\sum_{y_t\in\mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1})\cdot Q^{G_\theta}(Y_{1:t-1},y_t)],
\end{aligned}
$$

Algorithm

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
    for g-steps **do**
        Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
        for t in $1 : T$ **do**
            Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
        end for
        Update generator parameters via policy gradient Eq. (8)
    end for
    for d-steps **do**
        Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
        Train discriminator D_ϕ for k epochs by Eq. (5)
    end for
    $\beta \leftarrow \theta$
until SeqGAN converges

Detail

- Generator: RNN
- Dsicriminator: CharCNN
- Result:

Table 2: Chinese poem generation performance comparison

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Boundary Seeking Generative Adversarial Network

R Devon Hjelm, Athul Paul Jacob, Yoshua Bengio

- Use f-GAN formula
- Introduce importance sampling for discrete case

f-divergence family

- f-divergence family:

$$D_f(P\|Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x,$$

- f: $R_+ \rightarrow R$, f(1)=0

Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \mathrm{d}x$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \mathrm{d}x - \log(4)$	$u \log u - (u+1) \log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

Variational analysis on f-divergence

- Conjugate function (Convex)
  $$f^*(t) = \sup_{u \in dom_f} \{ut - f(t)\}$$
- $D_f(P||Q) = \int_X q(x) \sup_{t \in dom_{f^*}} \left\{ t\frac{p(x)}{q(x)} - f^*(t) \right\} dx$
- $\geq \sup_{T \in T} (\int p(x)\, T(x) dx \ - \int q(x) f^*(T(x)) dx)$
- $= \sup_{T \in T} (E_{\mathrm{P}}[T(x)] - E_Q[f^*(T(x))])$
- Which is a lower bound of the distribution difference
- The bound is tight if T can be any function
- Optimal $T^*(x) = f'(\frac{p(x)}{q(x)})$ if f,p,q has value (

Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)}\, dx$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)}\, dx$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\, dx$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 dx$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2}\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)}\, dx$	$-(u+1)\log\frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)}\, dx - \log(4)$	$u \log u - (u+1)\log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

GAN setting

- T: Discriminator

Theorem 1. *Let f be a convex function and $T^\star \in \mathcal{T}$ a function that satisfies the supremum in Equation 4 in the non-parametric limit. Let us assume that $\mathbb{P}$ and $\mathbb{Q}_\theta(x)$ are absolutely continuous w.r.t. a measure μ and hence admit densities, $p(x)$ and $q_\theta(x)$. Then the target density function, $p(x)$, is equal to $(\partial f^\star/\partial T)(T^\star(x))q_\theta(x)$.*

Proof. Following the definition of the f-divergence and the convex conjugate, we have:

$$\mathcal{D}_f(\mathbb{P}||\mathbb{Q}_\theta) = \mathbb{E}_{\mathbb{Q}_\theta}\left[f\left(\frac{p(x)}{q(x)}\right)\right] = \mathbb{E}_{\mathbb{Q}_\theta}\left[\sup_t\left\{t\frac{p(x)}{q(x)} - f^\star(t)\right\}\right]. \tag{6}$$

As $f^\star$ is convex, there is an absolute maximum when $\frac{\partial f^\star}{\partial t}(t) = \frac{p(x)}{q_\theta(x)}$. Rephrasing t as a function, $T(x)$, and by the definition of $T^\star(x)$, we arrive at the desired result. $\square$

Importance weight estimator

- In practice(not optimal case), the estimation may be biased
- Let $w^*(x) = \frac{\partial f^*(x)}{\partial T} T^*(x)$ (if tight, it equals $\frac{p(x)}{q(x)}$)
- Let $\beta = E_Q[w(x)]$
- $\hat{p}(x) = \frac{w(x)}{\beta} q(x)$ is an estimator of p(x)
- May be biased, but the bias only depends on the tightness of Variational lower bound

Discrete..

- In discrete case, we don't have $\nabla_x D(x)$
- But now we can use this importance sampling:

$$\nabla_\theta \mathcal{D}_{KL}(\tilde{p}(x)||q_\theta) = -\mathbb{E}_{\mathbb{Q}_\theta}\left[\frac{w(x)}{\beta}\nabla_\theta \log q_\theta(x)\right].$$

- This gradient equation allow us to train
- Estimating β has a high variance

Decrease variance

Lower-variance policy gradient Let $q_\theta(x) = \int_{\mathcal{Z}} g_\theta(x \mid z)h(z)dz$ be a probability density function with a conditional density, $g_\theta(x \mid z) : \mathcal{Z} \to [0,1]^d$ (e.g., a multivariate Bernoulli distribution), and prior over z, $h(z)$. Let $\alpha(z) = \mathbb{E}_{g_\theta(x|z)}[w(x)] = \int_{\mathcal{X}} g_\theta(x \mid z)w(x)dx$ be a partition function over the conditional distribution. Let us define $\tilde{p}(x \mid z) = \frac{w(x)}{\alpha(z)} g_\theta(x \mid z)$ as the (normalized) conditional distribution weighted by $\frac{w(x)}{\alpha(z)}$. The expected conditional KL-divergence over $h(z)$ is:

$$\mathbb{E}_{h(z)}[\mathcal{D}_{KL}\left(\tilde{p}(x \mid z) \| g_\theta(x \mid z)\right)] = \int_{\mathcal{Z}} h(z)\mathcal{D}_{KL}\left(\tilde{p}(x \mid z) \| g_\theta(x \mid z)\right) dz \tag{9}$$

Let $x^{(m)} \sim g_\theta(x \mid z)$ be samples from the prior and $\tilde{w}(x^{(m)}) = \frac{w(x^{(m)})}{\sum_{m'} w(x^{(m')})}$ be a Monte-Carlo estimate of the normalized importance weights. The gradient of the expected conditional KL-divergence w.r.t. the generator parameters, θ, becomes:

$$\nabla_\theta \mathbb{E}_{h(z)}[\mathcal{D}_{KL}\left(\tilde{p}(x \mid z) \| g_\theta(x \mid z)\right)] = -\mathbb{E}_{h(z)}\left[\sum_m \tilde{w}(x^{(m)}) \nabla_\theta \log g_\theta(x^{(m)} \mid z)\right], \tag{10}$$

where we have approximated the expectation using the Monte-Carlo estimate.

Algorithm

Algorithm 1 . Discrete Boundary Seeking GANs

$(\theta, \phi) \leftarrow$ initialize the parameters of the generator and statistic network
repeat
  $\hat{x}^{(n)} \sim \mathbb{P}$ ▷ Draw N samples from the empirical distribution
  $z^{(n)} \sim h(z)$ ▷ Draw N samples from the prior distribution
  $x^{(m|n)} \sim g_\theta(x \mid z^{(n)})$ ▷ Draw M samples from each conditional $g_\theta(x \mid z^{(m)})$ (drawn independently if $\mathbb{P}$ and $\mathbb{Q}_\theta$ are multi-variate)
  $w(x^{(m|n)}) \leftarrow (\partial f^\star / \partial T) \circ (\nu \circ F_\phi(x^{(m|n)}))$
  $\tilde{w}(x^{(m|n)}) \leftarrow w(x^{(m|n)}) / \sum_{m'} w(x^{(m'|n)})$ ▷ Compute the un-normalized and normalized importance weights (applied uniformly if $\mathbb{P}$ and $\mathbb{Q}_\theta$ are multi-variate)
  $\mathcal{V}(\mathbb{P}, \mathbb{Q}_\theta, T_\phi) \leftarrow \frac{1}{N} \sum_n F_\phi(\hat{x}^{(n)}) - \frac{1}{N} \sum_n \frac{1}{M} \sum_m w(x^{(m|n)})$ ▷ Estimate the variational lower-bound
  $\phi \leftarrow \phi + \gamma_d \nabla_\phi \mathcal{V}(\mathbb{P}, \mathbb{Q}_\theta, T_\phi)$ ▷ Optimize the discriminator parameters
  $\theta \leftarrow \theta + \gamma_g \frac{1}{N} \sum_{n,m} \tilde{w}(x^{(m|n)}) \nabla_\theta \log g_\theta(x^{(m|n)} \mid z)$ ▷ Optimize the generator parameters
until convergence

REINFORCE

- Can revise previous policy gradient equation using REINFORCE algorithm:

Definition 2.4 (REINFORCE-based BGAN). Let $T_\phi(x)$ be defined as above where $\partial f^\star / \partial T(T_\phi(x)) = e^{F_\phi(x)}$. Consider the gradient of the *reversed* KL-divergence:

$$\nabla_\theta \mathcal{D}_{KL}(q_\theta \| \tilde{p}) = -\mathbb{E}_{h(z)}\left[\sum_m (\log w(x^{(m)}) - \log \beta + 1) \nabla_\theta \log g_\theta(x^{(m)} \mid z)\right]$$
$$= -\mathbb{E}_{h(z)}\left[\sum_m (F_\phi(x) - b) \nabla_\theta \log g_\theta(x^{(m)} \mid z)\right] \qquad (11)$$

MaskGAN: Better Text Generation via Filling in the_______

William Fedus, Ian Goodfellow, Andrew M. Dai

- Task: Fill in the missing token
- Use Seq2seq generator
- Use Actor-critic

Motivation

- MLE method: Good perplexity, bad quality
- Tend to generate same word
- GAN -> More flexible, but not working directly
- Use filling the blank task to show this works better than traditional method

Actor-Critic

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

GAN

- Reward: log of discriminator output
- Training: REINFORCE

$$\nabla_\theta \mathbb{E}_G[R_t] = (R_t - b_t)\nabla_\theta \log G_\theta(\hat{x}_t)$$

- Critic b_t

Result

Ground Truth	**the next day 's show <eos> interactive telephone technology has taken a new leap in <unk> and television programmers are**
MaskGAN	the next day 's show <eos> interactive telephone technology has taken a new leap <u>in its retail business <eos> a</u>
MaskMLE	the next day 's show <eos> interactive telephone technology has taken a new leap <u>in the complicate case of the</u>

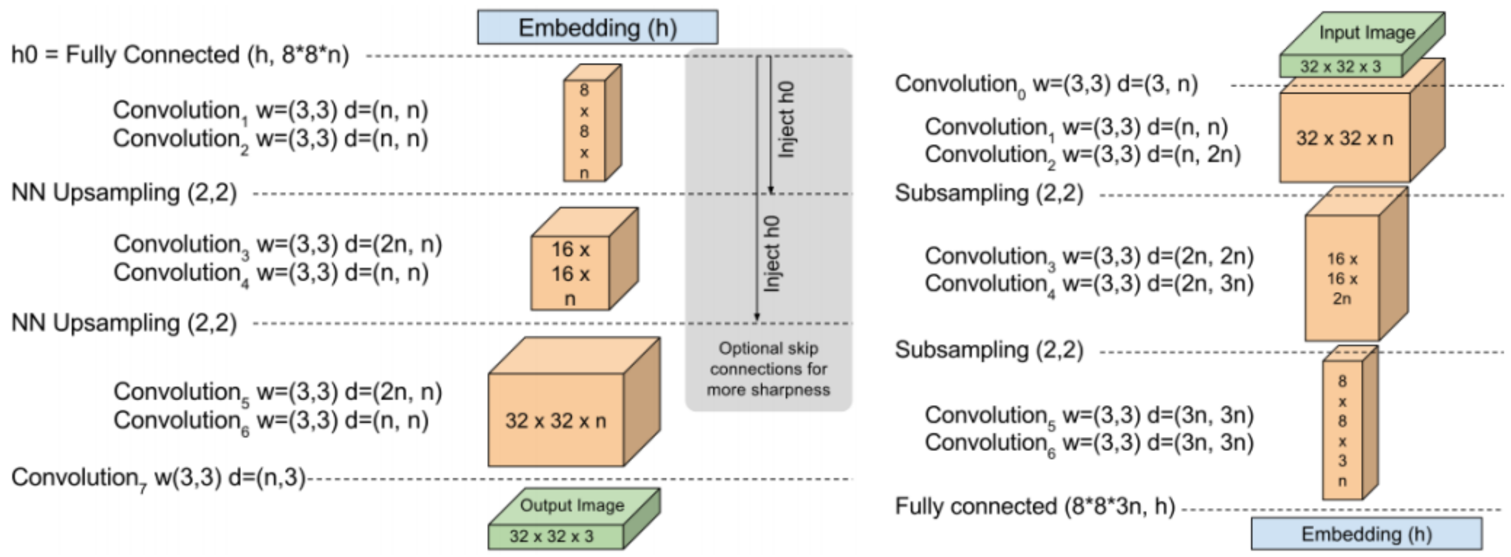
BEGAN: Boundary Equilibrium Generative Adversarial Networks

David Berthelot, Thomas Schumm, Luke Metz, Google 2017

- Use autoencoder as discriminator
- Use a new loss function (on a new target)
- Use a new GAN objective function: Boundary Equilibrium

Network

- Generator = Encoder
- Discriminator = Encoder + Decoder



(a) Generator/Decoder

(b) Encoder

Target: ~~Sample~~ Autoencoder Loss

- Minimize the Wasserstein distance between the autoencoder loss distribution instead of sample distribution:

We first introduce $\mathcal{L} : \mathbb{R}^{N_x} \mapsto \mathbb{R}^+$ the loss for training a pixel-wise autoencoder as:

$$\mathcal{L}(v) = |v - D(v)|^\eta \text{ where } \begin{cases} D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} & \text{is the autoencoder function.} \\ \eta \in \{1, 2\} & \text{is the target norm.} \\ v \in \mathbb{R}^{N_x} & \text{is a sample of dimension } N_x. \end{cases}$$

Let $\mu_{1,2}$ be two distributions of auto-encoder losses, let $\Gamma(\mu_1, \mu_2)$ be the set all of couplings of μ_1 and μ_2, and let $m_{1,2} \in \mathbb{R}$ be their respective means. The Wasserstein distance can be expressed as:

$$W_1(\mu_1, \mu_2) = \inf_{\gamma \in \Gamma(\mu_1, \mu_2)} \mathbb{E}_{(x_1, x_2) \sim \gamma}[|x_1 - x_2|]$$

Using Jensen's inequality, we can derive a lower bound to $W_1(\mu_1, \mu_2)$:

$$\inf \mathbb{E}[|x_1 - x_2|] \geqslant \inf |\mathbb{E}[x_1 - x_2]| = |m_1 - m_2| \tag{1}$$

Training of GAN

- m1: loss of real data
- Discriminator goal:

$$W_1(\mu_1, \mu_2) \geqslant m_2 - m_1$$
$$m_1 \to 0$$
$$m_2 \to \infty$$

Total objective: Use control theory to maintain equilibrium

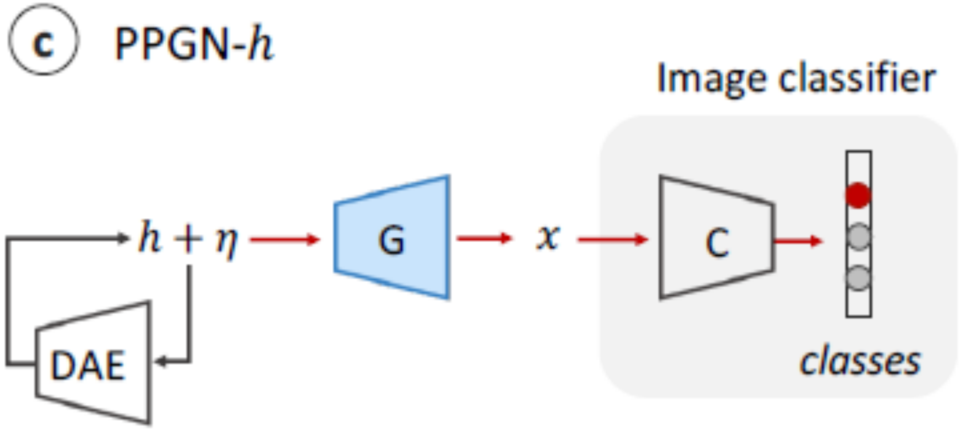
$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t.\mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma\mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases}$$

- γ is defined as

$$\gamma = \frac{\mathbb{E}\left[\mathcal{L}(G(z))\right]}{\mathbb{E}\left[\mathcal{L}(x)\right]}$$

Plug & Play Generative Networks: Conditional Iterative Generation of Images in latent space

Summary: Propose a type of generators: PPGN, which use GAN and a classifier together to generate better samples

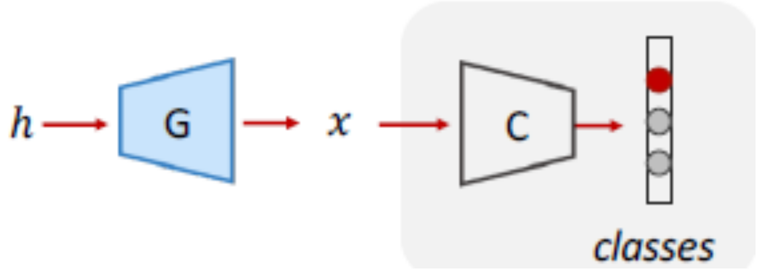


Previous work: DGN-AM

Optimize to find which h can highly activates a neuron in classifier C (i.e., activation maximization)

Such G can be transfered to other C, to produce other valid results

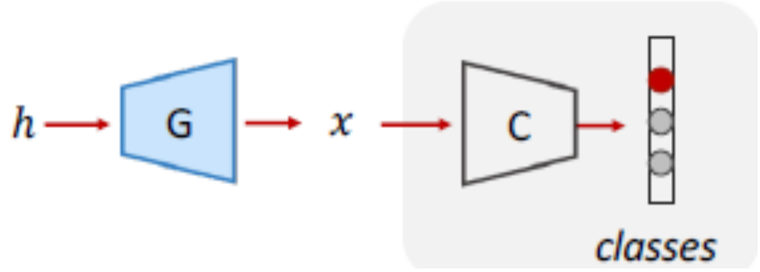
Issue: The images are too similar due to the activation maximization



Idea: Build a probabilistic framework for activation maximization

Sampling from a joint model (x:input, y:target label) can be split into two parts

$$p(x, y) = p(x)p(y|x)$$



Suppose we want to generate sample for class y_c

p(x): Generate good image

$p(y=y_c|x)$: Classified to be a certain class

Metropolis Hastings

For a distribution p(x), if we want to estimate it without an IID sampler, we need MCMC methods for sampling

M x) is a simple Gaussian
N

1. $x_{t+1} = x_t + N(0, \sigma^2)$
2. $\alpha = p(x_{t+1})/p(x_t)$
3. if $\alpha < 1$, reject sample x_{t+1} with probability $1 - \alpha$ by setting $x_{t+1} = x_t$, else keep x_{t+1}

In theory, it will produce samples for any computable p(x)

MALA(Metropolis-adjusted Langevin Algorithm)

Problem of MH:

1. Converge slow
2. We need p(x) to calculate alpha (Sometimes hard)

MALA is a revised algorithm:

1. $x_{t+1} = x_t + \sigma^2/2\nabla \log p(x_t) + N(0, \sigma^2)$
2. $\alpha = f(x_t, x_{t+1}, p(x_{t+1}), p(x_t))$
3. if $\alpha < 1$, reject sample x_{t+1} with probability $1 - \alpha$ by setting $x_{t+1} = x_t$, else keep x_{t+1}

MALA-Approx

Stochastic gradient Langevin dynamics can relax the requirement of exact p(x):

Simply use a stochastic gradient descent plus noise in the process:

$$x_{t+1} = x_t + \sigma^2/2\nabla \log p(x_t) + N(0, \sigma^2)$$

The real equation used in these algorithm:

Back to the problem:

$$x_{t+1} = x_t + \epsilon_{12} \nabla \log p(x_t | y = y_c) + N(0, \epsilon_3^2)$$
$$= x_t + \epsilon_{12} \nabla \log p(x_t) + \epsilon_{12} \nabla \log p(y = y_c | x_t) + N(0, \epsilon_3^2)$$

Another decoupling:

$$x_{t+1} = x_t + \epsilon_1 \frac{\partial \log p(x_t)}{\partial x_t} + \epsilon_2 \frac{\partial \log p(y = y_c | x_t)}{\partial x_t} + N(0, \epsilon_3^2)$$

Connection to previous activation maximization

$$x_{t+1} = x_t + \epsilon_1 \frac{\partial \log p(x_t)}{\partial x_t} + \epsilon_2 \frac{\partial \log p(y = y_c | x_t)}{\partial x_t} + N(0, \epsilon_3^2)$$

Activation Maximization with no prior: (e1,e2,e3) = (0,1,0)

Gaussian prior: Use (e1,e2,e3) = (lambda, 1, 0)

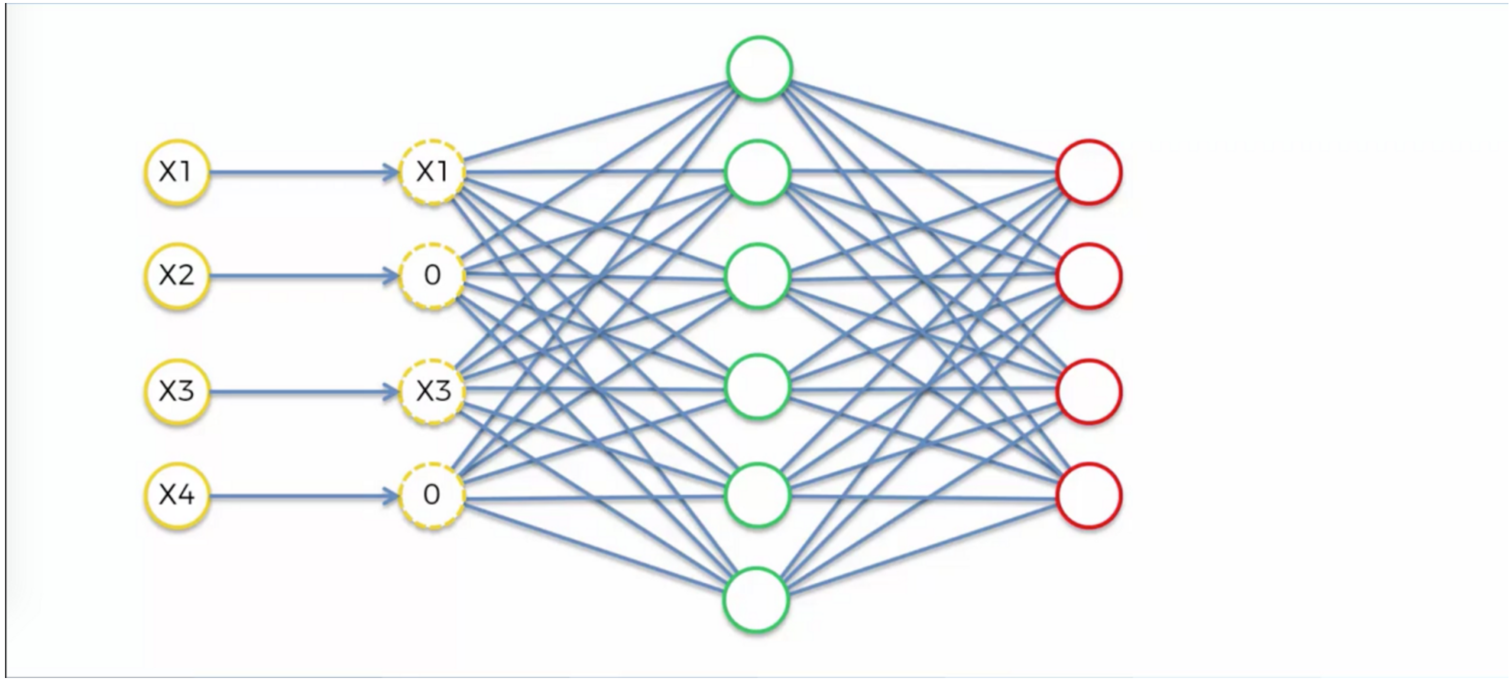
Hand driven prior: Add a new regularization term in the equation

Previous alogirthm doesn’t have the noise term, therefore, easy to generate similar image

Denoising autoencoder

Use a denoising autoencoder to provide the prior

Denoising autoencoder: Add some noise in the hidden representation, try to do the reconstruction

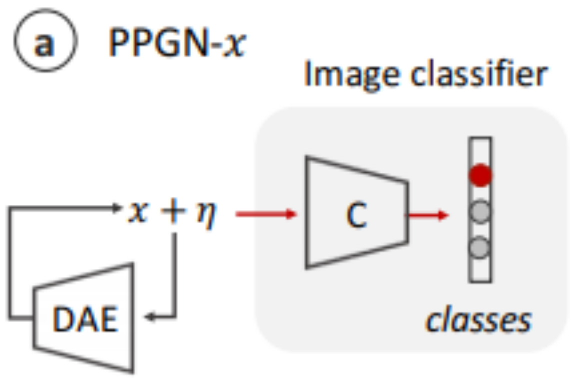


Model I

DAE + Classifier

Use DAE to model x and produce x

Sampling from the whole model

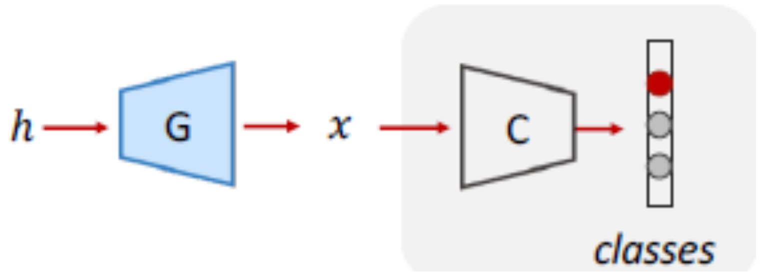


Problem: Performance is bad

DGN-AM

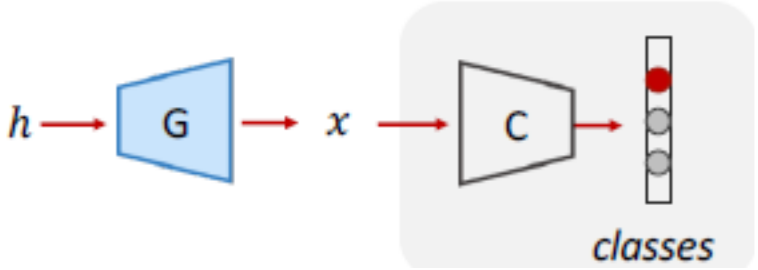
Use GAN to model x

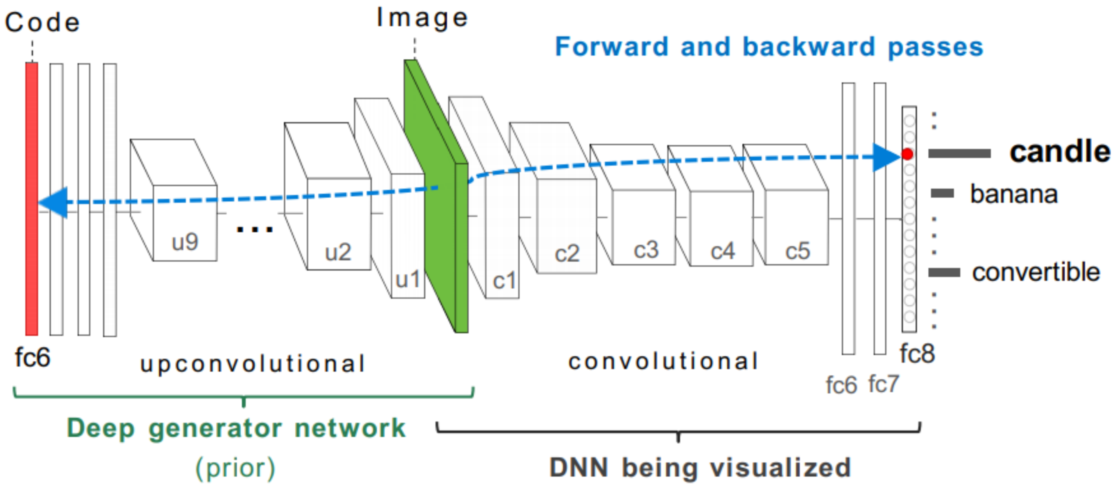
Optimize to find which h can highly activates a neuron in classifier C (i.e., activation maximization)



DGN-AM

- 4 CNNs:
- 1) a fixed encoder network E to be inverted
- 2) a generator network G
- 3) a fixed “comparator” network C
- 4) a discriminator D
- G is trained to invert a feature representation extracted by the network E. Satisfy 3 objectives:
- 1. For a feature vector yi = E(xi), the synthesized image G(yi) has to be close to the original image xi 2. The features of the output image C(G(yi)) have to be close to those of the real image C(xi)
- 3. D should be unable to distinguish G(yi) from real images.

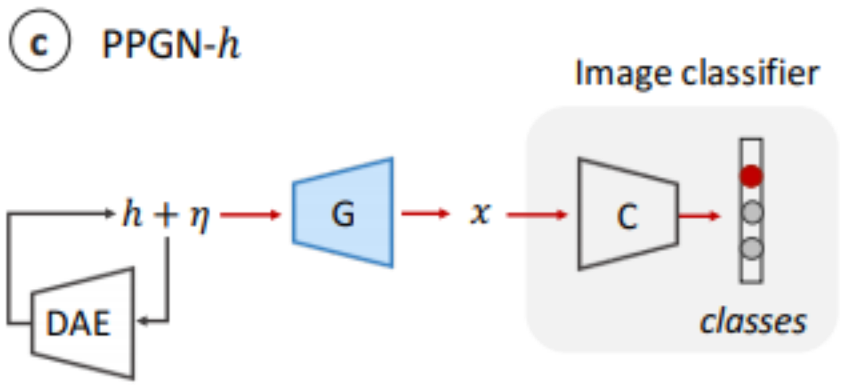




PPGN-h

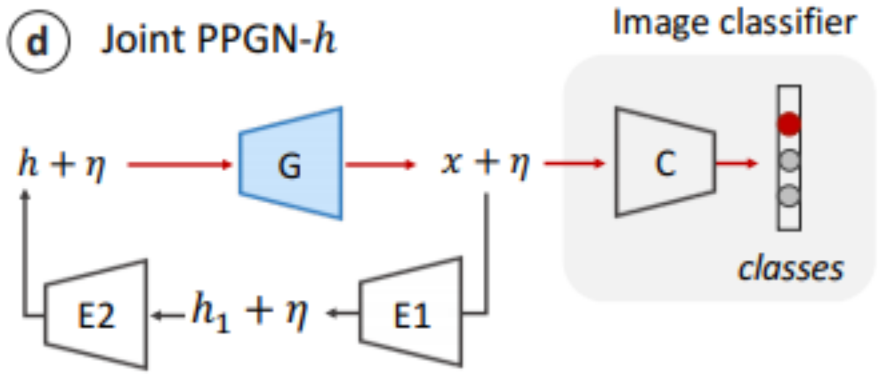
Use GAN + DAE

DAE: Produce better prior for sampling



Joint PPGN-h

Use multiple DAEs, for a better reconstruction of the prior



Noiseless Joint PPGN-h

Doesn't use noise, get better performance in practice...

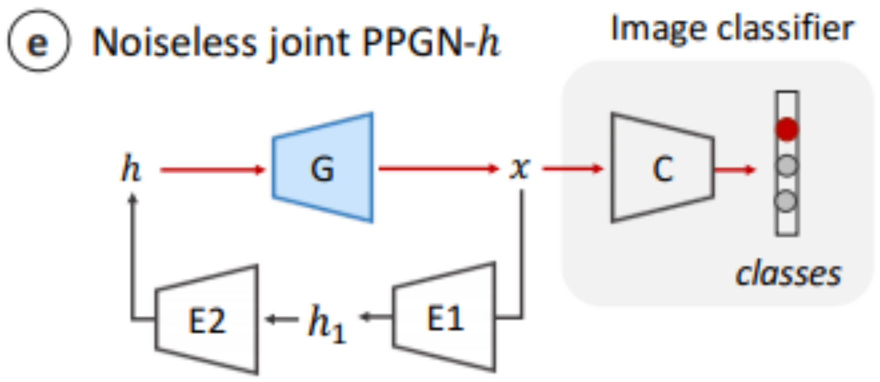
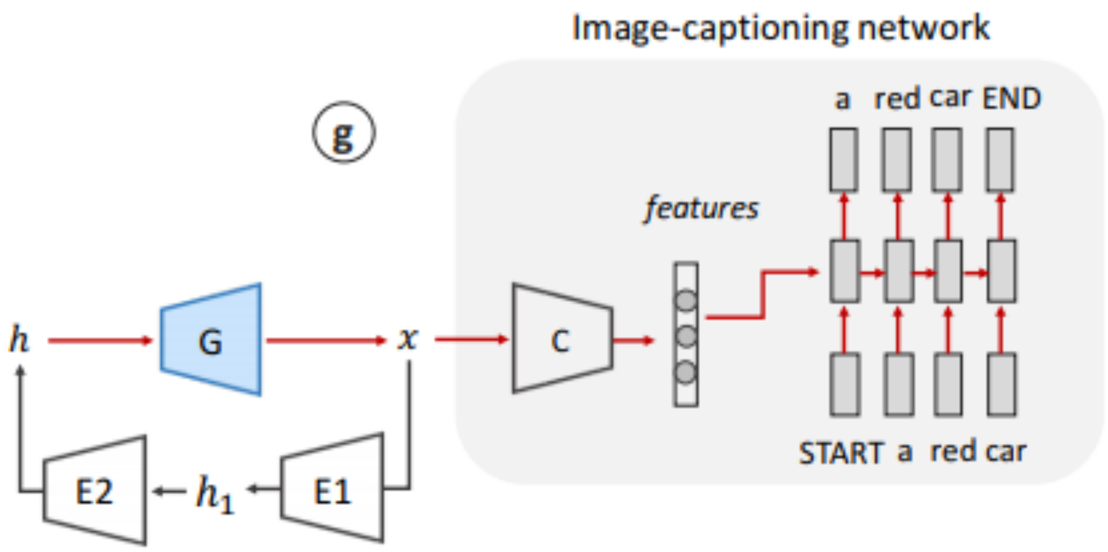


Image captions

Can be used generate image by image captions



Sampling conditioning on **captions**

Experiment result

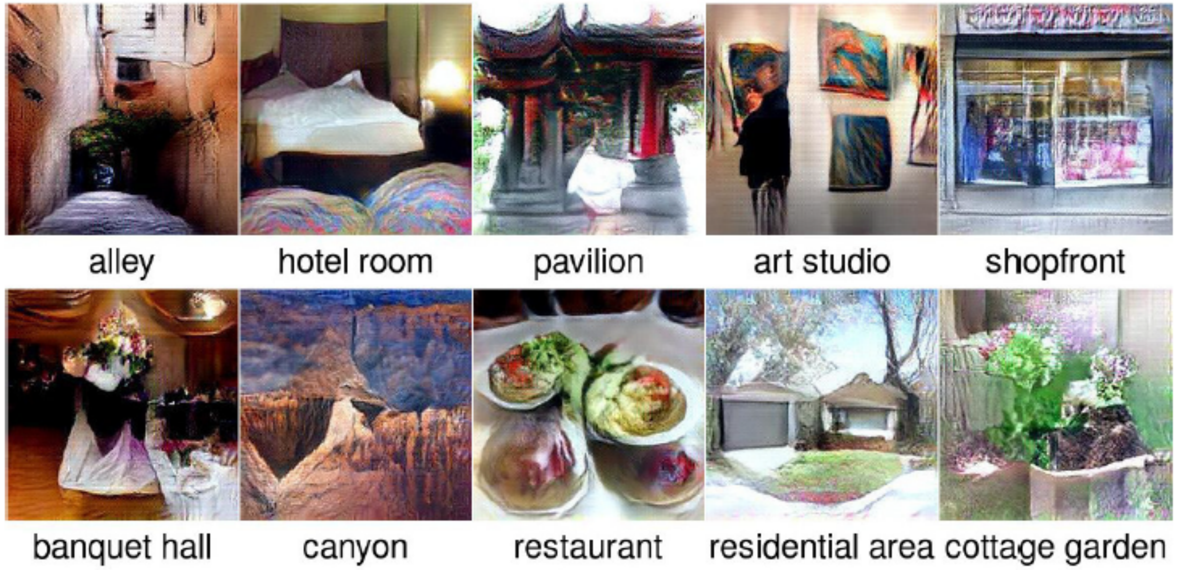


Figure 4: Images synthesized conditioned on MIT Places [65] classes instead of ImageNet classes.

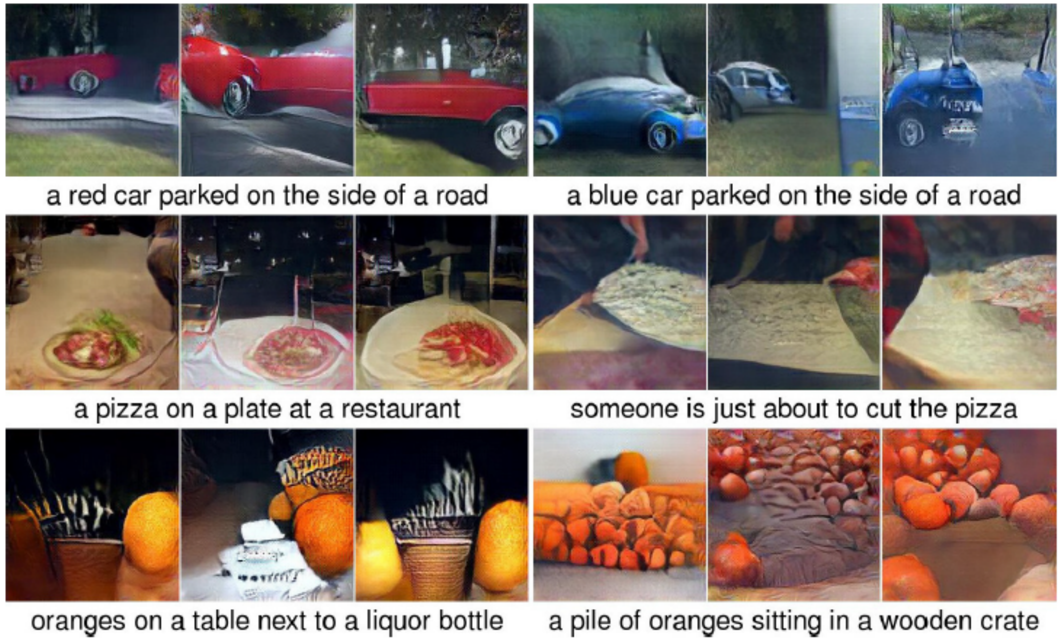


Figure 5: Images synthesized to match a text description. A PPGN containing the image captioning model from [8] can generate reasonable images that differ based on user-provided captions (e.g. *red* car vs. *blue* car, oranges vs. *a pile* of oranges). For each caption, we show 3 images synthesized starting from random codes (more in Fig. S18).