

生成对抗模型调研报告

丁铭

August 18, 2017

Abstract

传统意义的生成模型是通过观测到的数据来刻画整个分布, 获得分布任意点处的值的模型。然而, 最近兴起了另一种“生成模型”, 它们虽然不能得到整个分布的具体值, 但是可以按照该分布产生样本。这样的模型被称为“隐式生成模型”。在这当中比较成功的是生成对抗网络 (Generative Adversarial Network)。另外在生成序列信息的时候, 强化学习的思想往往能取得很好的成果。本文将通过部分论文介绍生成对抗模型的相关知识。

Contents

1 概述	3
2 传统生成模型	3
2.1 Auto-Encoding Variational Bayes[19]	3
2.2 Generating Sequences With Recurrent Neural Networks[15]	4
2.3 Photographic Image Synthesis with Cascaded Refinement Networks[6]	5
3 GAN 理论	6
3.1 原始 GAN	6
3.1.1 Generative Adversarial Networks[14]	6
3.2 WGAN	7
3.2.1 Towards Principled Methods for Training Generative Adversarial Networks[1]	7
3.2.2 Wasserstein GAN[2]	8
3.2.3 Improved Training of Wasserstein GANs[16]	8
3.3 LS-GAN	9
3.3.1 Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities[29]	9
3.4 基于能量函数的 GAN 的变体	9
3.4.1 Energy-based Generative Adversarial Network[39]	9
3.4.2 BEGAN: Boundary Equilibrium Generative Adversarial Networks[5] . .	12
3.5 GAN 的问题	13
3.5.1 Do GANs actually learn the distributions? An empirical study[3]	13
4 GAN 实践	13
4.1 高质量图片生成	13

4.1.1	Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[30]	13
4.1.2	Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks[11]	13
4.1.3	Generating Images Part by Part with Composite Generative Adversarial Networks[20]	14
4.1.4	Generating images with recurrent adversarial networks[17]	16
4.1.5	StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks[37]	16
4.2	pix2pix 风格转换	16
4.2.1	Image-to-Image Translation with Conditional Adversarial Networks[18]	16
4.2.2	Unsupervised Cross-Domain Image Generation[33]	18
4.2.3	Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks[40]	18
4.2.4	One-Sided Unsupervised Domain Mapping[4]	20
4.2.5	Generative Semantic Manipulation with Contrasting GAN[23]	20
4.3	生成控制	21
4.3.1	Conditional Image Synthesis With Auxiliary Classifier GANs[27]	21
4.3.2	InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets[7]	21
4.4	调参技巧	22
4.4.1	Improved Techniques for Training GANs[31]	22
5	GAN 应用	23
5.1	半监督学习	23
5.1.1	Semi-Supervised Learning with Generative Adversarial Networks[26]	23
5.1.2	Triple Generative Adversarial Nets[22]	23
5.1.3	Good Semi-supervised Learning That Requires a Bad GAN[10]	24
5.2	Multi-View Image Generation from a Single-View[38]	24
5.3	Adversarial Training For Sketch Retrieval[9]	24
5.4	Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network[21]	26
5.5	SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient[36]	26
5.6	Style Transfer Generative Adversarial Networks: Learning to Play Chess Differently[8]	27
5.7	SEGAN: Speech Enhancement Generative Adversarial Network[28]	29
6	强化学习	29
6.1	Playing Atari with Deep Reinforcement Learning[24]	29
6.2	Human-level control through deep reinforcement learning[25]	29
6.3	Deep Reinforcement Learning with Double Q-learning[34]	33
6.4	Dueling Network Architectures for Deep Reinforcement Learning[35]	33

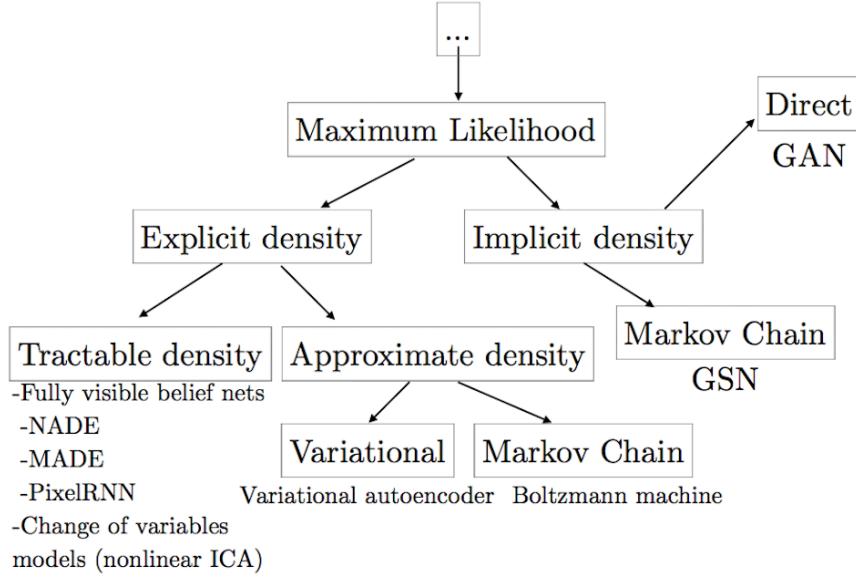


Figure 1: 生成模型分类

1 概述

在 GAN 发明人, Ian Goodfellow 在 NIPS2016 的讲稿 [13] 中, 他曾将生成模型如图 1 分类。可以看出 GAN 只是生产模型中新兴起的小分支。

本文主要介绍 GAN 的原理、进展及应用, 但是也将在正式介绍之前适当介绍部分其他生成模型。

GAN 的理论发展主要集中在损失函数上, 比较有价值的几个进展是 :

1. GAN, 优化生成网络的 JS 散度
2. WGAN, 优化生成网络的 Wasserstein 距离
3. LSGAN, 机器学习确定损失函数

这些是本文的重点。

GAN 的实践中总结出了很多技巧和模型, 它们在图像生成任务中可以取得更好的效果。这样的工作最多, 文章大部分的篇幅将围绕它们展开。

虽然 GAN 最初出现在图像领域, 但是很多人将 GAN 应用于 NLP、IR 等其他领域, 并取得了不错的效果。文章也将选择性地介绍几篇应用的文章。

文章的最后将介绍几篇强化学习的论文, 在 GAN 的应用, 特别是在序列数据生成方面, 很多问题和强化学习中的问题是相似的, 越来越多的任务使用了强化学习中的技巧。

2 传统生成模型

2.1 Auto-Encoding Variational Bayes[19]

Variational AutoEncoder 是一个传统的生成模型, 它处理如下传统问题:

存在样本 $x^{(i)} \in R^M$, 假设样本分布 $P(x)$ 可以由如下两步生成 :

- 生成隐变量 $z \sim p_\theta(z)$, $z \in R^K$

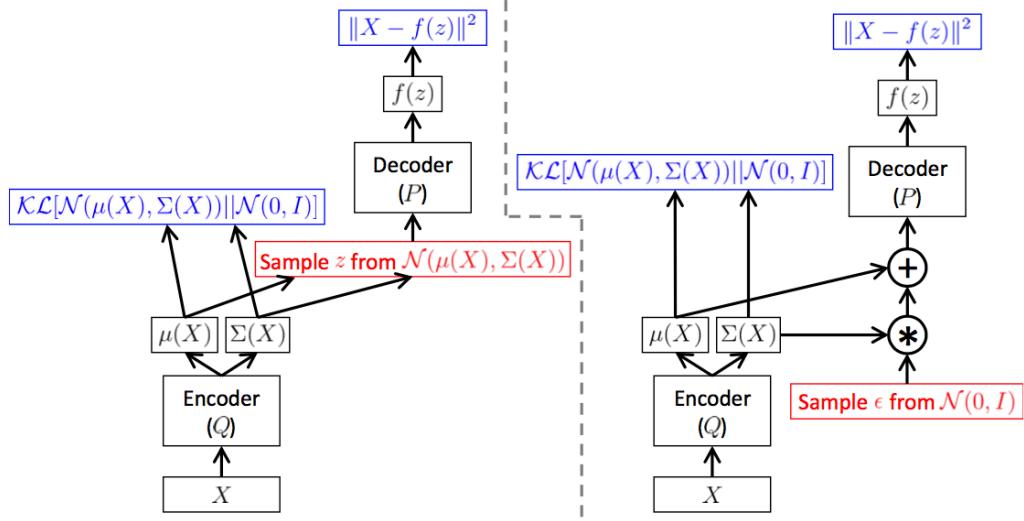


Figure 2: VAE

- 生成样本 $x \sim p_\theta(x|z)$

同时由于 x 边缘似然难以积分 (intractable)，所有隐变量的后验概率 $p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$ 是难以计算的。不能使用 EM 算法。

从 $D_{KL}(q(z|X)||p(z|X)) = -\mathbb{E}_{z \sim q(z|X)}[\log \frac{p(z|X)}{q(z|X)}]$ 可以变形推出：

$$\log p(X) = D_{KL}(q(z|X)||p(z|X)) + L(p, q, X)$$

$$L(p, q, X) = -D_{KL}(q(z|X)||p(z)) + \mathbb{E}_{z \sim q(z|X)}[\log p(X|z)]$$

由于 KL 散度大于 0， L 被称为 variational lower bound。

我们的目标就是优化 $q(z|X)$ 的参数 ϕ 和 $p(X|z)$ 的参数 θ ，同时假设 z 的先验概率为 $N(0, I)$ 。根据 q 、 p 模型的不同，可以得到不同的 VAE，文中给出的例子是二者均为多元高斯分布。

现在面临的另一个问题是，刚才的推断建立在针对单一样本 X 上。即使我们得到了一组最优的参数，可是它们只是条件概率的参数，对于其他样本来说仍然不适用。所以我们更希望得到的是 $Q(X) \rightarrow \phi P(z) \rightarrow \theta$ 这两个用神经网络去拟合。其实这里已经与假设有点不同了，假设的生成过程中 θ 与 z 应该是无关的，而这里变成了 z 的函数。

但是网络上大部分的写法与论文差距更大——直接用神经网络拟合 $p(X|z)$ ，可是 X 通常很高维且连续，神经网络不可能生成连续分布，于是将模型转化为隐式生成模型，并将 sample 得到的值同标准值的 MSE 作为 loss 的第二部分。

这种做法的原理未能查到任何出处。但是 [12] 以图2的形式展示了这一做法。

2.2 Generating Sequences With Recurrent Neural Networks[15]

这是通过 RNN、LSTM 等模型直接生成序列的方法，通过 MLE 进行训练。对于某个样

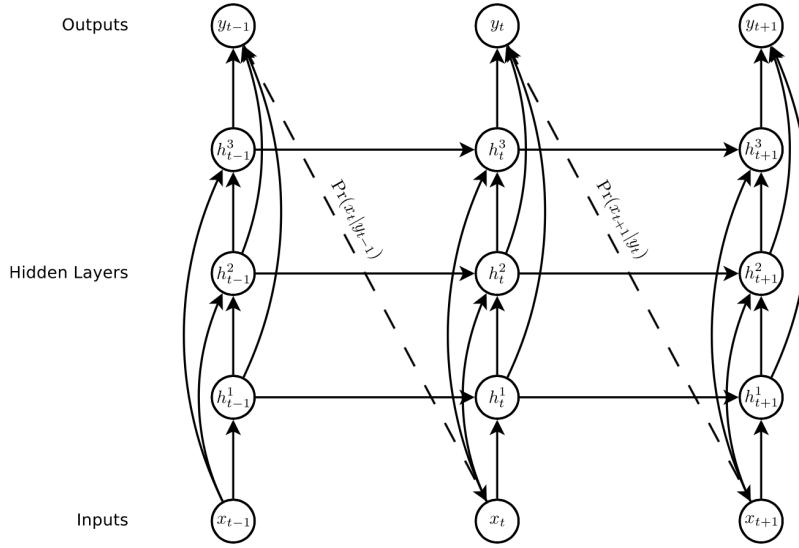


Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

Figure 3: RNN 直接生成序列

本,

$$loss = - \sum_{t=1}^T \log Pr(x_{t+1}|y_t)$$

具体做法是 :

1. predict 的时候 x_{-1} 为 0, 预测 y_0 , 它是一个在字典上的条件概率分布, 然后根据这个分布随机生成一个作为 x_0 输入到下一层,
2. train 的时候使用上面的公式, 每次输入数据中的元素, 同时计算这个元素在模型中的生成概率, 也就是 y_t 中的对应项。

这里有个小 trick, 据说训练 LSTM 的时候梯度可能非常大, 在 LSTM 的 input 处对梯度做截断。

2.3 Photographic Image Synthesis with Cascaded Refinement Networks[6]

这是一篇与 pix2pix 的工作, 针对其中的 mask 转化为真实图像的任务。文章使用了大量数据, 没有采用 GAN 而是一种比较直接的卷积神经网络生成, 取得了 state-of-art 的效果。

这说明监督学习中, 如果有大量成对的数据, 可能还是直接映射的方法比较好。文章输入 mask 图片, 并且不断经过多个 Module, 每次分辨率长宽翻倍。每个 Module 分三层, 输入上层的输出和相应大小的 mask 图片, 输出一定 channels (256) 的图片。效果图见4。注意最后的 loss 是生成图像、真实图像通过 VGG 网络各层之间的差距 $\sum_l \lambda_l ||\Phi_l(I) - \Phi(g_u(L, \theta))||_1$ 。

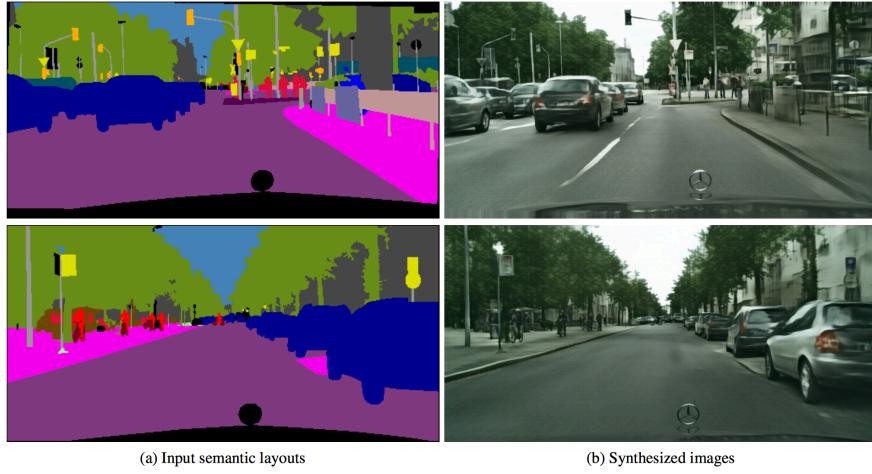


Figure 4: photographic 效果图

3 GAN 理论

3.1 原始 GAN

3.1.1 Generative Adversarial Networks[14]

如果我们抛开传统模型，重新考虑如何建立一个隐式的生成模型，最容易想到的是将高斯噪音通过神经网络，生成样本这样的模型。可是我们很快就发现没有办法训练它——贝叶斯方法很难用来训练神经网络参数，因为连 $p(\mathcal{D}|\theta)$ 都无法确定。传统的神经网络通过梯度下降来优化损失函数，可是生成样本来自随机噪音并不能找到配对的真实样本。于是我们会去想将 loss 定义为与数据集中最近样本之间的距离（我并不确定这样效果如何，但找最近样本本身就很费时的）等全局的方法，它们通常都不太优美。但是 GoodFellow 做到了，GAN 应运而生。

GAN 优化两个网络——生成网络 G 和判别网络 D 。生成网络通过噪音生成“假”样本，而判别网络判别样本的真假。损失函数为

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

首先看从整个取值空间来看，

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z}))) d\mathbf{z} \\ &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \end{aligned} \quad (1)$$

如果我们充分优化 D ，对于每一个固定的 \mathbf{x} ，被积项中 $D(\mathbf{x})$ 都能取到 $[0, 1]$ 上的极大值 $\frac{p_{\text{data}}}{p_{\text{data}} + p_g}(a \log x + b \log(1 - x))$ 求导。之后优化，

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{P_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_g} \left[\log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] \\ &= -\log(4) + KL \left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2} \right) + KL \left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2} \right) \\ &= -\log(4) + 2 \cdot JSD(p_{\text{data}} \| p_g) \end{aligned}$$

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```

for number of training iterations do
  for  $k$  steps do
    • Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
    • Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution
       $p_{\text{data}}(\mathbf{x})$ .
    • Update the discriminator by ascending its stochastic gradient:
```

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)})))].$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

于是 Loss 函数唯一最小值时有 $p_g = p_{\text{data}}$ 。并且可以证明如果优化能在整个函数空间上进行，那么可以收敛。

3.2 WGAN

3.2.1 Towards Principled Methods for Training Generative Adversarial Networks[1]

这篇文章是 WGAN 前作，指出了 GAN 的问题所在。实际训练为什么要“平衡”生成和判别模型，而原文说直接将判别模型训练到最好就可以了。原文指出 D 训练到最优时，

$$D(x) = \frac{p_{\text{data}}(x)}{p_g(x) + p_{\text{data}}(x)}$$

此时对于生成模型的优化，

$$L(D^*, g_\theta) = 2JSD(\mathbb{P}_g || \mathbb{P}_{\text{data}}) - 2\log 2$$

只需优化（梯度下降）这个 JS 散度。但实际上，这个梯度下降难以进行。假设 p_{data} p_g 重合度很低，此时发现 JS 散度是常数，没办法梯度下降。实际上他们都是高维空间（图像 $n*m$ ）的低维流形，不重合概率很大。原文提出的另一种改变 G 的 loss 则受梯度不稳定和 mode collapse 的困扰。

这篇文章给出的方法是给噪音退火，使得初值达到一个有不少重合的状态。

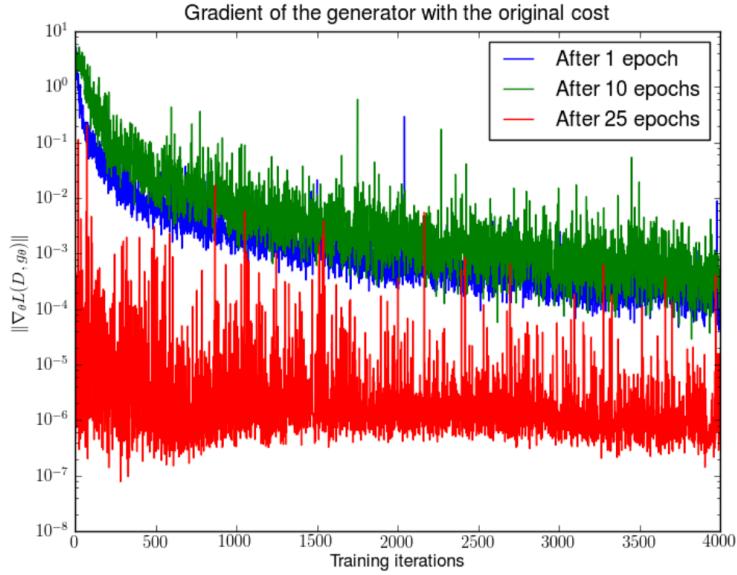


Figure 5: 训练时间与梯度

3.2.2 Wasserstein GAN[2]

Wasserstein 距离 $W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(x,y) \sim \gamma} [|x - y|]$ 就是找一个分配方案，使得联合概率分布集中在对角线附近。一阶距就是 Earth-Mover。分布没有重叠的时候 Wasserstein 距离也有梯度。有定理

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim P_r} [f(x)] - \mathbb{E}_{x \sim P_g} [f(x)]$$

即对于所有满足 Lipschitz 条件的且 Lipschitz 常数小于 K 的函数中，期望差的上界。证明略。用一个神经网络代替 f 的集合。为了满足 Lipschitz 常数小于 K，强行将权值 clip 到一个规定的范围内。这样得到的值是 Wasserstein 距离的常数倍，梯度方向不变。

这个距离应该是 G 的 loss，那么就用这个神经网络 f 代替 D，使用样本充分训练使这个值极大后就代表了 p_g p_r 之间的 Wasserstein 距离。那么最后实际更改的就只有：

- 判别器最后一层去掉 sigmoid
- 生成器和判别器的 loss 不取 log
- 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数 c
- 不要用基于动量的优化算法（包括 momentum 和 Adam），推荐 RMSProp，SGD 也行

3.2.3 Improved Training of Wasserstein GANs[16]

weight clipping 有两个重要问题：

1. 倾向于最大最小值，可能减小网络表达能力
2. 容易梯度爆炸或消失

效果不太好。于是不直接限制梯度，而是在后面加一个惩罚项。将判别器的目标改为

$$L = \mathbb{E}_{x \sim p_g} [D(x)] - \mathbb{E}_{x \sim p_r} [D(x)] + \lambda \mathbb{E}_{x \sim p_x} [(||\nabla_x D(x)||_2 - 1)^2], \lambda = 10$$

需要用到二阶梯度，可以用差分代替。

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α , the learning rate. c , the clipping parameter. m , the batch size. n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```

1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ 
12: end while
```

Figure 6: WGAN 算法描述

3.3 LS-GAN

3.3.1 Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities[29]

GAN 优化分布之间的 JS 散度, WGAN 优化分布之间的 Earth-Mover 距离, 而这篇文章通过 Lipschitz 假设认为这个 loss-function 可以通过真假样本间的距离之间学习, 虽然我们没有成对的数据, 但是由于 Lipschitz 条件, 可以定性地认为真实分布集中在几个区域, 那么我们可以将噪音分别映射到这些区域。我们定义 L_θ 是一个对于真实样本输出较小值, 对于假样本输出较大值的函数, 同时满足限制

$$L_\theta(G(z)) - L_\theta(x) \geq \Delta(x, G(z))$$

其中 Δ 为样本距离比如一阶距。所以我们可以轮流优化两个网络。

优化 Loss function, $\mathbb{E}_{x \sim p_{\text{data}}} L_\theta(x) + \lambda \mathbb{E}_{x \sim p_{\text{data}}, z \sim p_z} (\Delta(x, G(z)) - L_\theta(G(z)) + L_\theta(x))$ 优化 G , $\mathbb{E}_{z \sim p_z} L_\theta(G(z))$ 。算法描述如9。

3.4 基于能量函数的 GAN 的变体

3.4.1 Energy-based Generative Adversarial Network[39]

文章重定义了 GAN 的度量方法, 假设存在一个能量函数, 它在真实样本处接近于 0, 在假样本处接近于 m。那么我们定义 G, D 的 Loss 为 :

$$\mathcal{L}_D(x, z) = D(x) + \max(m - D(G(z)), 0)$$

$$\mathcal{L}_G(z) = D(G(z))$$

那么这个系统的 Nash 均衡点可以证明是

$$p_g = p_{\text{data}}$$

是得到。定义 loss-function 为 autoencoder 的重构误差[10], 认为这样做好处是梯度来源提供的信息量比较大。另外文章定义了一个 pull-away term, 让每个 batch 生成的有差别的一个



Figure 7: 效果比较

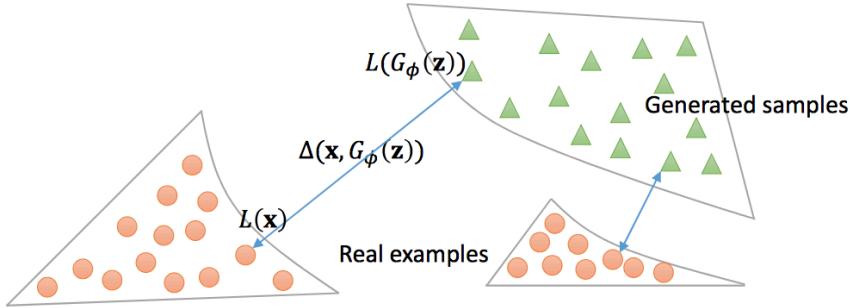


Fig. 1. Illustration of the idea behind LS-GAN. A margin is enforced to separate real samples from generated counterparts. The margin is not fixed to a constant. Instead it is data-dependent, which could vanish as the generator improves to produce better and better samples. We assume the density of real samples is Lipschitz as to prove the theoretical results.

Figure 8: LS-GAN 图示

Algorithm 1 Learning algorithm for LS-GAN.

```

Input:  $m$  data examples  $\mathcal{X}_m$ , and  $\lambda$ .
for a number of iterations do
    for a number of steps do
        \\\ Update the loss function over minibatches;
        Sample a minibatch from  $\mathcal{X}_m$ ;
        Sample a minibatch from  $\mathcal{Z}_m$ ;
        Update the loss function  $L_\theta$  by descending the gradient of (8) with weight decay over the minibatches;
    end for
    Sample a set of  $\mathcal{Z}'_k$  of  $k$  random noises;
    Update the generator  $G_\phi$  by descending the gradient of (9) with weight decay;
    Update the generated samples  $\mathbf{x}^{(m+j)} = G_{\phi^*}(\mathbf{z}_j)$  for  $j = 1, \dots, m$  on  $\mathcal{Z}_m$ ;
end for

```

Figure 9: LS-GAN 算法

$$D(x) = \|Dec(Enc(x)) - x\|. \quad (13)$$

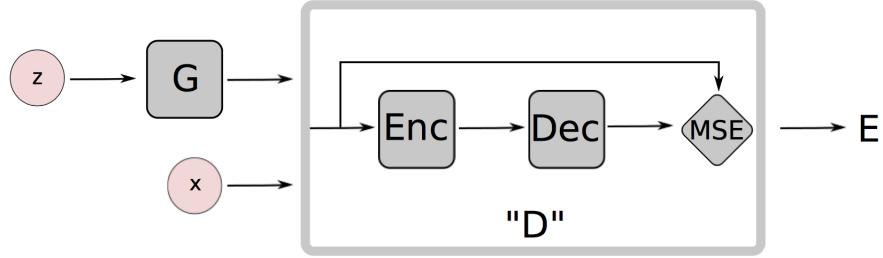


Figure 10: EBGAN 图解

The BEGAN objective is:

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

Figure 11: BEGAN 训练过程

loss 项。

$$\frac{1}{N(N-1)} \sum_i \sum_{j \neq i} \left(\frac{\mathbf{S}_i^T \mathbf{S}_j}{\|\mathbf{S}_i\| \|\mathbf{S}_j\|} \right)^2$$

\mathbf{S} 为 encode 出来的向量。

3.4.2 BEGAN: Boundary Equilibrium Generative Adversarial Networks[5]

这篇文章是基于 EBGAN 的，首先修改了均衡的假设，不再是真实样本为 0，生成样本为 m，而是满足一个接近 1 的倍数关系

$$\gamma \mathcal{L}(x) = \mathcal{L}(G(z))$$

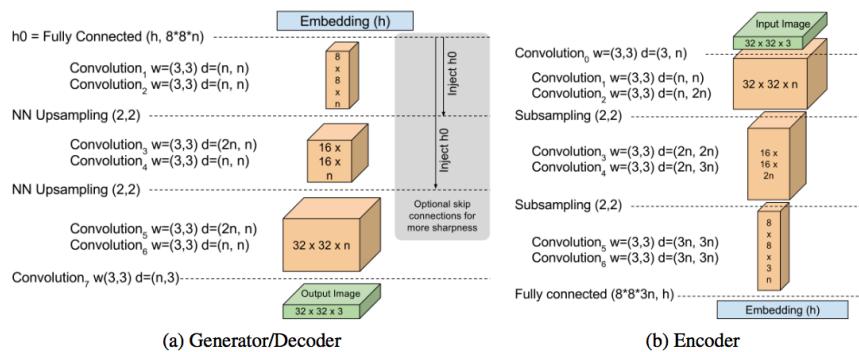


Figure 12: EBGAN 图解

Theorem 1. Given a discrete probability distribution P on a set Ω , if there exists a subset $S \subseteq \Omega$ of size N such that $\sum_{s \in S} P(s) \geq \rho$, then the probability of encountering at least one collision among M i.i.d. samples from P is $\geq 1 - \exp(-\frac{m^2\rho}{2N})$

Theorem 2. Given a discrete probability distribution P on a set Ω , if the probability of encountering at least one collision among M i.i.d. samples from P is γ , then for $\rho = 1 - o(1)$, there exists a subset $S \subseteq \Omega$ such that $\sum_{s \in S} P(s) \geq \rho$ with size $\leq \frac{2M\rho^2}{(-3 + \sqrt{9 + \frac{24}{M} \ln \frac{1}{1-\gamma}}) - 2M(1-\rho)^2}$

Figure 13: 定理

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Figure 14: 设计建议

3.5 GAN 的问题

3.5.1 Do GANs actually learn the distributions? An empirical study[3]

作者使用实验来说明 GAN 学出来的分布 support size (离散分布中的非 0 点) 不大, 这是一个连续分布, 用类 (mode) 代替。根据生日悖论, 如果我们能够随机抽一些看重复, 可以估算出分布的 support size。

这样估计出了大部分概率构成的分布的 support size 的上界。实验证明学得的分布的 support size 远小于应有的数据集的 support size。证明了 mode collapse 的严重程度。

4 GAN 实践

4.1 高质量图片生成

4.1.1 Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[30]

文章仅仅是将 MLP 用卷积神经网络实现了一下, 但是通过细致的调参产生了很好的效果。[14](#)给出了一些设计的建议, [15](#)是最终使用的网络架构。

4.1.2 Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks[11]

图像的 laplacian 金字塔在 SIFT 中用到过, 但是这里似乎只用到了上下采样的层次, 没有每个 octave 内部的分辨率变化。于是我们训练多个 GAN, 直接生成最小的, 然后 conditional

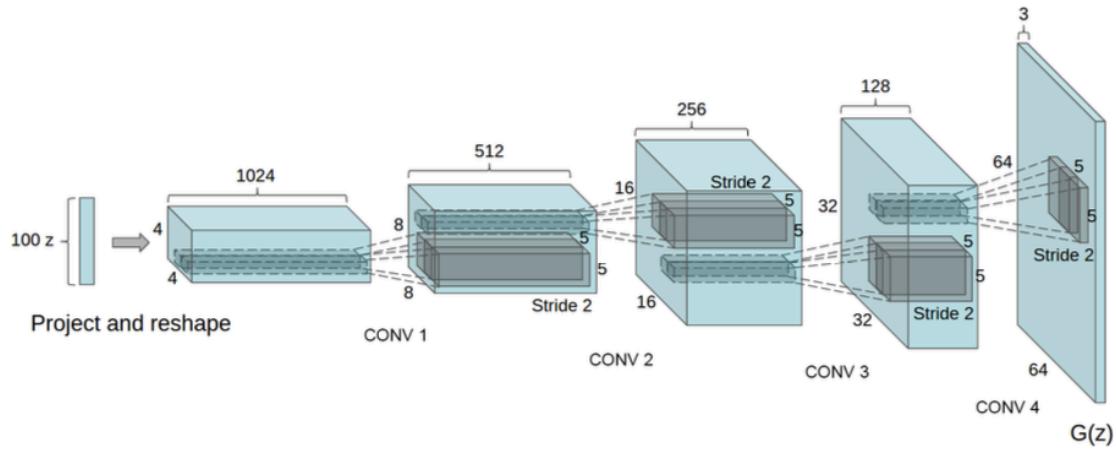


Figure 15: 生成器架构

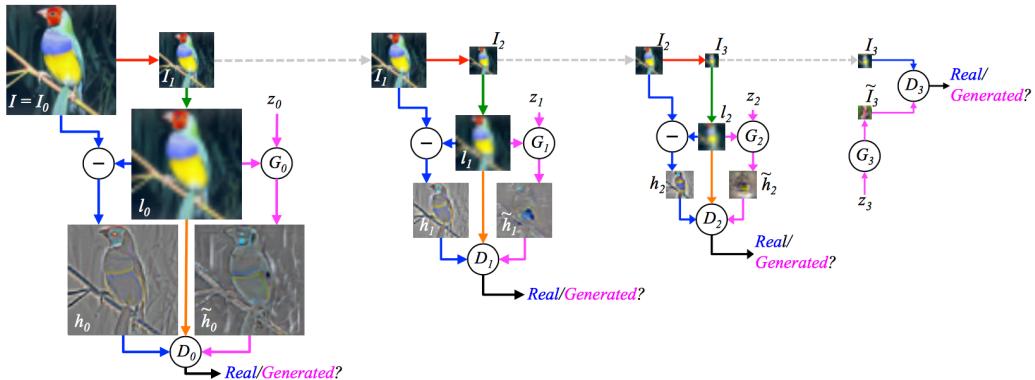


Figure 16: LAPGAN 架构

GAN 的方法将 upsampled 的图像输进去生成差值。

解决“GAN 直接生成图像一旦分辨率较大就出问题”。

4.1.3 Generating Images Part by Part with Composite Generative Adversarial Networks[20]

想法是将图片生成问题分为多部分，比如背景、前景、装饰等……使用一个 RNN 产生时序的输出，然后通过不同的 G 来生成不同的部分。各部分通过 alpha-blend 融合在一起，就是将后一张图片中透明权重用前面生成的图片代替。为了防止只有一个 G 起作用，增加了一个正则

$$|u - \sum C_{ijA}| - \sum (C_{ijA} - 0.5)^2$$

其中 u 是一个自己定义的不透明度之和，防止前面都生成透明图像。后面的强迫透明度取 0、1。

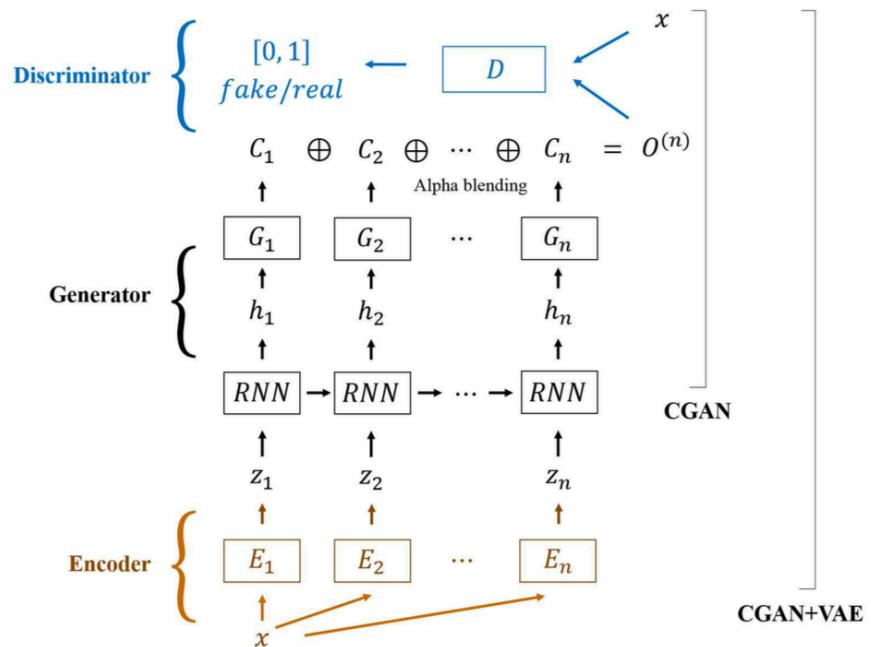


Figure 2: The structure of CGAN and CGAN+VAE. The discriminator and generator are main two components of CGAN. The RNN accepts a noise sequence z_1, z_2, \dots, z_n as input, and then recursively generates h_1, h_2, \dots, h_n which are then passed through each generator independently creating images C_1, C_2, \dots, C_n having RGBA channels. The images are then combined sequentially by the alpha blending process to form the final output $O^{(n)}$. Note that generators do not share weights, meaning that they are all different networks. We additionally combined variational autoencoder (VAE) to create z_1, z_2, \dots, z_n directly from the images.

Figure 17: Composite GAN 架构

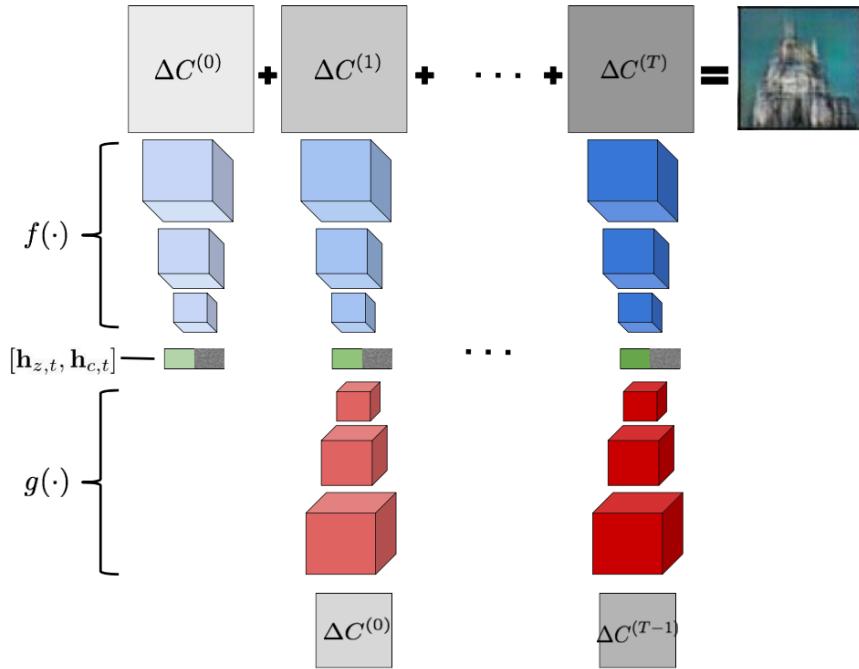


Figure 18: GRAN 架构

4.1.4 Generating images with recurrent adversarial networks[17]

学习 DRAW 的生成方法，分部分生成，感觉结构与 composite GAN 很像。文章使用了一个很 novel 的度量方法，但是总之效果并没有很好。

文章将 RNN 每层换成了 encoder-decoder 结构，但是隐含层的状态并没有传入下一层。其中 decoder 生成图片，是一个 DCGAN 的生成器。

4.1.5 StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks[37]

精心设计了两个 GAN 的架构，根据文本描述生成 256*256 的高质量图片。值得注意的是，文本输入并没有直接使用 embedding 向量，作者认为向量过于稀疏效果不好，于是将向量通过两个神经网络生成一个 N 元高斯的均值向量和协方差矩阵，并加一项正则

$$\lambda D_{KL}(\mathcal{N}(\mu(\phi_t), \Sigma(\phi_t)) || \mathcal{N}(0, I))$$

使得这个分布尽量接近标准正态。

4.2 pix2pix 风格转换

4.2.1 Image-to-Image Translation with Conditional Adversarial Networks[18]

文章讨论如何将监督信息融入到 GAN 中，增加与真实转化图片的距离。给判别器和生成器的输入都增加转化前的原图。

$$L_{L1}(G) = E_{x,y \sim p_{data}(x,y), z \sim p_z(z)} [\|y - G(x, z)\|_1]$$

$$L = L_{GAN}(G, D) + \lambda L_{L1}(G)$$

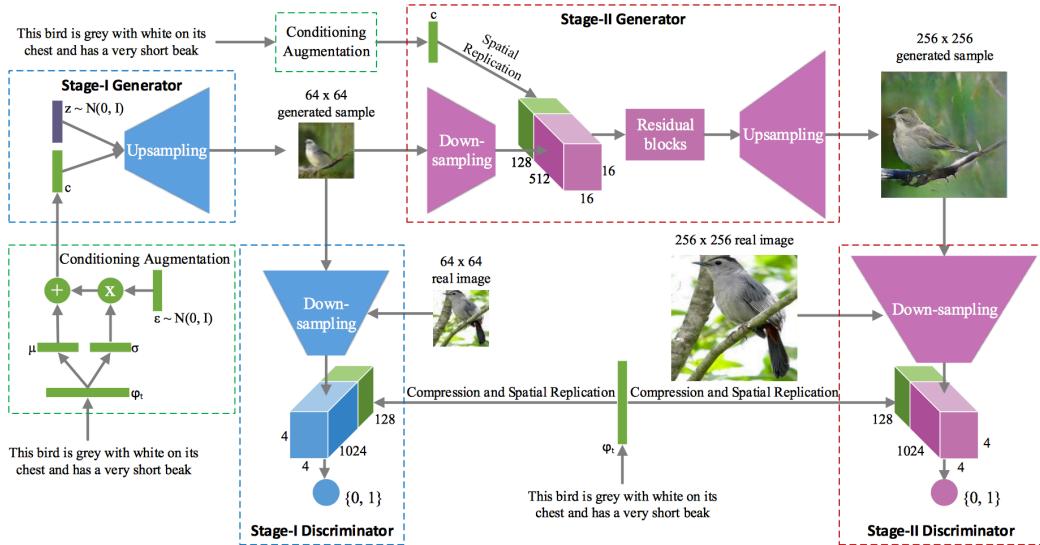


Figure 19: StackGAN 架构

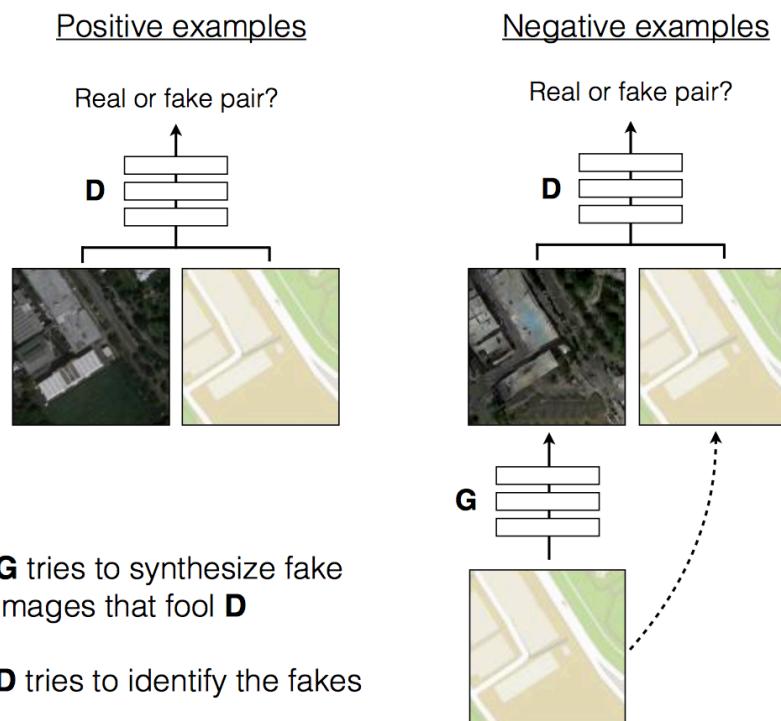


Figure 20: Conditional GAN

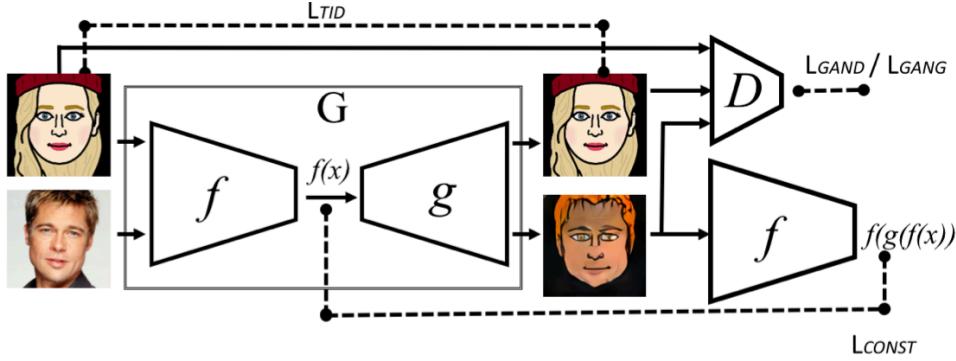


Figure 1: The Domain Transfer Network. Losses are drawn with dashed lines, input/output with solid lines. After training, the forward model G is used for the sample transfer.

Figure 21: 设计架构

生成模型中使用了 U-Net 的结构, 保证压缩信息的同时有原来的全部信息输入, 这个与 resNet 有相同之处。

4.2.2 Unsupervised Cross-Domain Image Generation[33]

研究这样一个问题 : 存在两个不同的 Domain S、T, 学习 $G(s) \rightarrow t$ 。同时有一个定义在 S 并 T 上的函数 f , 使得经过 G 的变化之后 f 不变, f 可能代表了整体架构之类的信息。精心设计了复杂的网络²¹, 结合了 auto-encoder (获得 f) 以及 GAN。

共多项损失²², 但是 lossTV 最后没加。

4.2.3 Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks[40]

还有两篇几乎同时发出的文章——Dual GAN、Disco GAN, 做法相同。

Question 之前 conditional GAN 的一个应用是 pix2pix, 从一种类型的图片转化为相同框架的另一种类型的图片。但是需要配对的数据, 很多时候我们并没有这样的数据, 只有两类数据的样本若干。

Method 通过重构误差。准确的说 cycle consistency, 这是一种历史悠久的技术了。具体见图²³。文章中完全取消了 conditional GAN 的随机性 (反正也没什么用)。

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log D_Y(G(x))]$$

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1]$$

实际将 log 换成平方效果更好。

仍然是对于 style 变化有用, 对于有 geometry 变化的东西效果不好, 如 dog->cat。

The second change alters the form of L_{GAN} , making it multiclass instead of binary. It also introduces a new term L_{TID} that requires G to be the identity matrix on samples from T . Taken together and written in terms of training loss, we now have two losses L_D and $L_G = L_{\text{GANG}} + \alpha L_{\text{CONST}} + \beta L_{\text{TID}} + \gamma L_{\text{TV}}$, for some weights α, β, γ , where

$$L_D = -\mathbb{E}_{x \in s} \log D_1(g(f(x))) - \mathbb{E}_{x \in t} \log D_2(g(f(x))) - \mathbb{E}_{x \in t} \log D_3(x) \quad (3)$$

$$L_{\text{GANG}} = -\mathbb{E}_{x \in s} \log D_3(g(f(x))) - \mathbb{E}_{x \in t} \log D_3(g(f(x))) \quad (4)$$

$$L_{\text{CONST}} = \sum_{x \in s} d(f(x), f(g(f(x)))) \quad (5)$$

$$L_{\text{TID}} = \sum_{x \in t} d_2(x, G(x)) \quad (6)$$

and where D is a ternary classification function from the domain T to 1, 2, 3, and $D_i(x)$ is the probability it assigns to class $i = 1, 2, 3$ for an input sample x , and d_2 is a distance function in T . During optimization, L_G is minimized over g and L_D is minimized over D . See Fig. 1 for an illustration of our method.

The last loss, L_{TV} is an anisotropic total variation loss (Rudin et al., 1992; Mahendran & Vedaldi, 2015), which is added in order to slightly smooth the resulting image. The loss is defined on the generated image $z = [z_{ij}] = G(x)$ as

$$L_{\text{TV}}(z) = \sum_{i,j} \left((z_{i,j+1} - z_{i,j})^2 + (z_{i+1,j} - z_{i,j})^2 \right)^{\frac{B}{2}}, \quad (7)$$

where we employ $B = 1$.

Figure 22: 损失函数

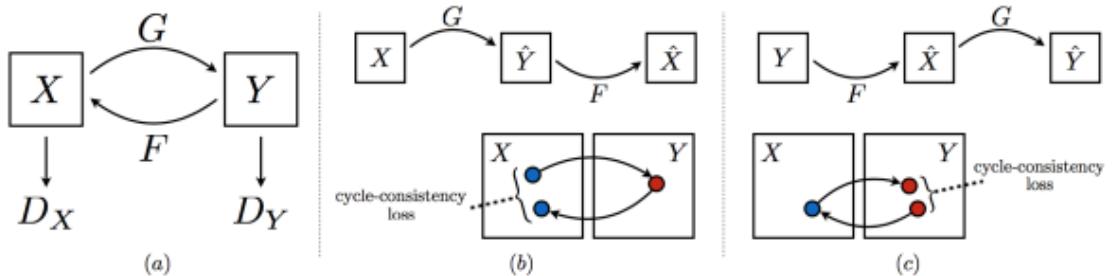


Figure 23: Cycle GAN 原理

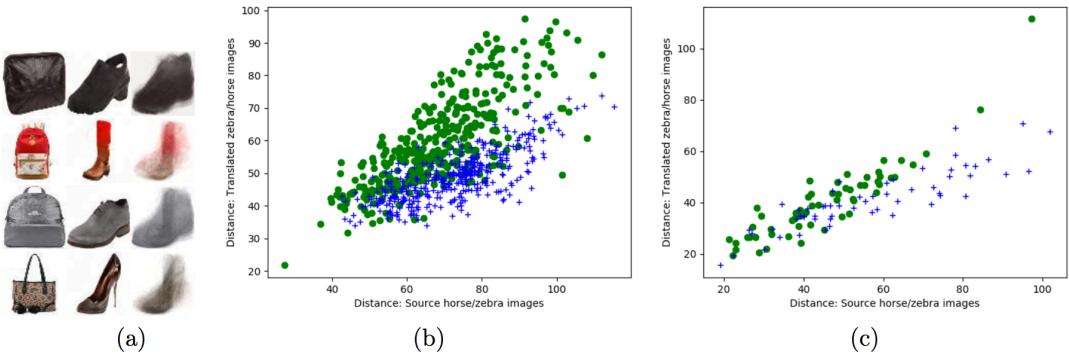


Figure 24: Distance GAN 效果

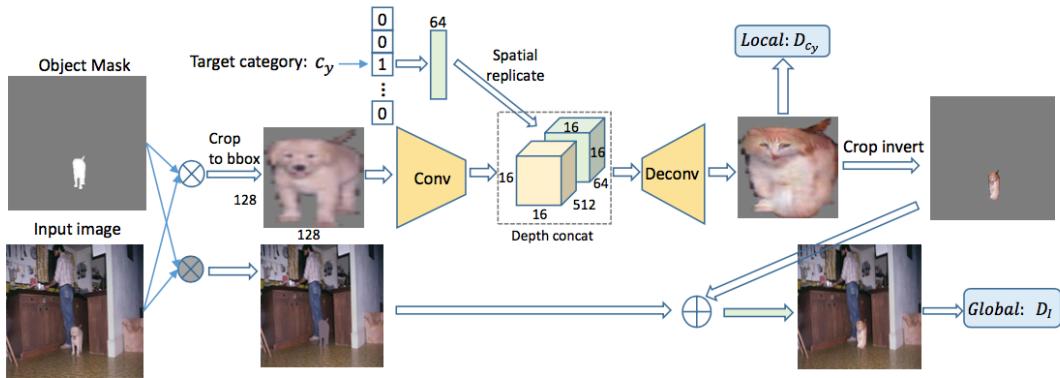


Figure 25: mask-conditional contrast-GAN

4.2.4 One-Sided Unsupervised Domain Mapping[4]

这是一篇改进 Cycle 效果的文章。将原来的 cycle 只保留一半，然后加一项正则。

$$L_{distance}(G_{AB}, \hat{p}_A) = \mathbb{E}_{x_i, x_j \sim \hat{p}_A} \left| \frac{1}{\sigma_A} (\|x_i - x_j\|_1 - \mu_A) - \frac{1}{\sigma_B} (\|G_{AB}(x_i) - G_{AB}(x_j)\|_1 - \mu_B) \right|$$

就是说原来 domain 的文章变换过去要归一化保距，这就更加强行限制了少出现 mode collapse。

4.2.5 Generative Semantic Manipulation with Contrasting GAN[23]

文章针对 cycle GAN 不能大幅改变物体的特点，提出用一个 mask 先找出需要替换的物体，然后用 GAN 生成另一类物体强行替换进去。[25](#)

GAN 生成的时候基本是基于 Cycle GAN，但是加入一项 contrast loss。

$$Q(f_{y'}, f_x, f_y) = -\log \frac{e^{-\|f_{y'} - \bar{f}_y\|}}{e^{-\|f_{y'} - \bar{f}_y\|} + e^{-\|f_{y'} - f_x\|}}$$

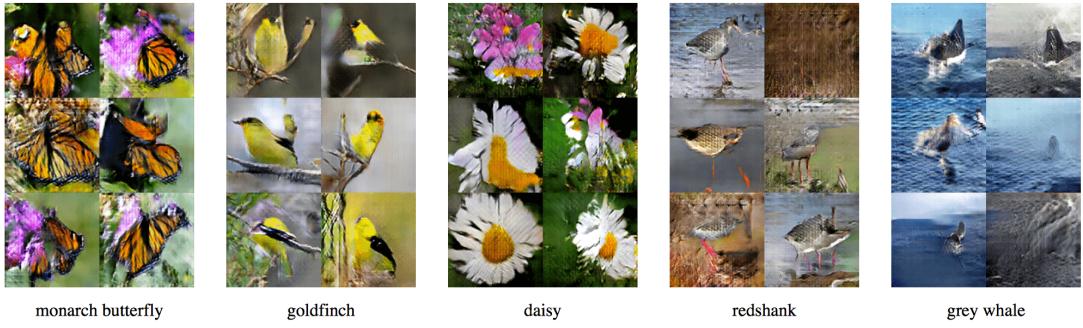


Figure 1. 128 × 128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet dataset. Note that the classes shown have been selected to highlight the success of the model and are not representative. Samples from all ImageNet classes are linked later in the text.

Figure 26: AC-GAN 效果

4.3 生成控制

4.3.1 Conditional Image Synthesis With Auxiliary Classifier GANs[27]

利用监督信息生成特定类图片。做法就是 G 输入噪音和类别信息，加入第二个 D 判别类别，这里类别判断都要最大化 likelihood。

$$L = L_{GAN}(G, D_1) + \mathbb{E}_{x \sim c_i} [\log D_2(x)_i]$$

后一项包括真假两种情况，按照输入 G 时的 c 来定。

4.3.2 InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets[7]

Question G 学习的是噪音分布到数据分布的映射，之前研究发现噪音向量有类似于 word2vec 的线性性质，但是具体想控制某个特征很难。希望能够让输入向量是 disentangled representation 的表示。

Method 信息论概念 确定一部分控制输入 c，加 regularization，使得已知生成样本后，对输入 c 的互信息很大。由于 c 是我们控制的，优化 G 只能将多个差异较大的样本分配给不同的 c，这样就可以显式控制了。

$$\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$$

由于不知道真实分布，更不知道条件分布，没办法计算互信息，可以通过自己构造的模型代替 p，并且把条件概率改为 c 的先验概率。

$$I(c; G(z, c)) \approx \mathbb{E}_{c \sim P(c), x \sim G(z, c)} [\log Q(c|x)] + H(c)$$

Implementation 新增加一部分网络 Q，Q 是将 D 前面的提取特征层重用，最后改为输出对 c 的条件概率的层。对于 c 取离散值的通过 softmax，c 取连续值的使用 factored Gaussian。

0 1 2 3 4 5 6 7 8 9	7 7 7 7 7 7 7 7 7 7
0 1 2 3 4 5 6 7 8 7	0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9	7 7 7 7 7 7 7 7 7 7
0 1 2 3 4 5 6 7 8 9	9 9 9 9 9 9 9 9 9 9
0 1 2 3 4 5 6 7 8 9	8 8 8 8 8 8 8 8 8 8

(a) Varying c_1 on InfoGAN (Digit type)(b) Varying c_1 on regular GAN (No clear meaning)

1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1
8 8 8 8 8 8 8 8 8 8	8 8 8 8 8 8 8 8 8 8
3 3 3 3 3 3 3 3 3 3	3 3 3 3 3 3 3 3 3 3
9 9 9 9 9 9 9 9 9 9	9 9 9 9 9 9 9 9 9 9
5 5 5 5 5 5 5 5 5 5	5 5 5 5 5 5 5 5 5 5

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure 27: InfoGAN 效果

4.4 调参技巧

4.4.1 Improved Techniques for Training GANs[31]

Question GAN 可以看成是找博弈的 Nash equilibrium, 但是并没有什么算法适合这种高维连续的参数空间。通过梯度下降不能保证博弈的收敛性, 比如 loss 为 xy , 两边都会在正负两边循环而不能收敛到 $(0,0)$ 。文中介绍几个 heuristic 的方法缓解这个问题。

feature matching 优化 G 只看结果偶然性较大, 更精细一点是比较生成样本和真实样本在 D 中某个中间层的特征表示。令 f 为 D 到这个中间层的映射。

$$J^{(G)} = \|\mathbb{E}_{x \sim data} f(x) - \mathbb{E}_{x \sim noise} f(G(x))\|_2^2$$

Minibatch discrimination 其实是一种层将 $n \times A$ 的中间层映射为 $n \times B$ 且将相互之间差异信息糅杂进去。具体做法是用 $A \times B \times C$ 的权值矩阵去乘, $out_{n,b} = \sum_{i=1}^N e^{-||(in_i T)_b - (in_n T)_b||_1}$ 事实证明这个不影响 D 的分类效果。

Historical averaging 加正则与历史平均参数值的差 $\|\theta - \frac{1}{t} \sum \theta[i]\|$ 。

One-sided label smoothing 用 0.1、0.9 代替 0、1。避免在截止区出现一系列问题。semi-supervised K+1 类, 其中 1 类代表 fake。分两段分别学习。

Inception Score

$$e^{\mathbb{E}_x D_{KL}(p(y|x) || p(y))}$$

x 为生成样本, y 为类别。



Figure 1. Output samples from SGAN and GAN after 2 MNIST epochs. SGAN is on the left and GAN is on the right.

Figure 28: Semi-Supervised GAN 效果

5 GAN 应用

5.1 半监督学习

5.1.1 Semi-Supervised Learning with Generative Adversarial Networks[26]

这是 Improve GAN 中提及的半监督学习的 GAN 的方式的出处，就是对于原来有分类的数据，将 D 的输入分成 N+1 类，这样可以同时做生成和判别两个任务。效果主要在收敛效率上，这两个任务在初期或样本少的时候使用 SGAN 效果都比单独的 Baseline 要好。

5.1.2 Triple Generative Adversarial Nets[22]

这篇文章想同时解决两个问题：半监督的分类和已知种类标签的生成。

前者之前是通过增加判别器种类实现的，后者在 AC-GAN 中是通过 classifier 的 loss 实现的。

本文将 generator、classifier、discriminator 三者更紧密的结合在一起，判别器接受 sample、label 对判别是否真实，而在这中 sample、label 对可能是真实 sample+classifier 生成的或者 generator 根据 label 生成的（本文中 generator 是 conditional 的）。最后加上真实数据和标签训练 classifier 的 loss。

$$\begin{aligned} & \min_{C,G} \max_D E_{(x,y) \sim p(x,y)} [\log D(x,y) - \log p_c(y|x)] + \\ & (1-\alpha) E_{y \sim p(y), z \sim p(z)} [\log(1 - D(G(y,z), y))] + \alpha E_{x \sim p(x), y \sim p_c(y|x)} [\log(1 - D(x, y))] \end{aligned}$$

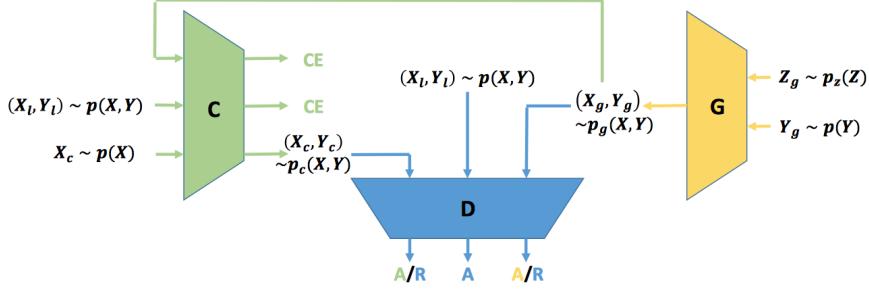


Figure 1: An illustration of Triple-GAN (best view in color). The utilities of D , C and G are colored in blue, green and yellow respectively, with “R” denoting rejection, “A” denoting acceptance and “CE” denoting the cross entropy loss for supervised learning. “A’s” and “R’s” are the adversarial losses and “CE’s” are unbiased regularizations that ensure the consistency between p_g , p_c and p .

Figure 29: Triple GAN

5.1.3 Good Semi-supervised Learning That Requires a Bad GAN[10]

kimi 的一篇文章，他发现之前直接在 D 加入分成 N 类的方法理论上有很大问题。设想如果是完美的生成器，那么 D 将无法确定如何究竟应该分在 $K+1$ 类还是生成的类别。事实上，GAN 做半监督之所以有用是因为会生成一些真实分布不同 mode 之间的类别，从而让分类器将这些分辨开，使得这中间低密度区域两边的分类更加准确。文章认为 G 产生的梯度应该更接近于真实分布的补集，于是提出优化 G 与 mode 的凸集内概率为 $\frac{1}{Z} \frac{1}{p(x)}$ 这个分布的 KL 散度，但为了保证其他地方没有分布，否则 KL 散度没法算，还得加上原来的 loss。³⁰

最后的 loss 是

$$\min_G -\mathcal{H}(p_G) + \mathbb{E}_{x \sim p_G} \log p(x) \mathbb{I}[p(x) > \epsilon] + \|\mathbb{E}_{x \sim p_G} f(x) - \mathbb{E}_{x \sim \mathcal{U}} f(x)\|^2.$$

第一项熵用类似与 INFO GAN 中的方法估计 $-\mathcal{H}(p_G) \leq -\mathbb{E}_{x, z \sim p_G} \log q(z|x) = L_{VI}$ ，第二项用 pixelCNN++[32] 的方法估计。最后一项使用原来的损失函数，又增加一个熵使得分布尽量偏向某个类。

$$\begin{aligned} \max_D & \mathbb{E}_{x, y \sim \mathcal{L}} \log p_D(y|x, y \leq K) + \mathbb{E}_{x \sim \mathcal{U}} \log p_D(y \leq K|x) + \\ & \mathbb{E}_{x \sim p_G} \log p_D(K+1|x) + \mathbb{E}_{x \sim \mathcal{U}} \sum_{k=1}^K p_D(k|x) \log p_D(k|x). \end{aligned} \quad (2)$$

5.2 Multi-View Image Generation from a Single-View[38]

这篇文章是通过 conditional GAN 框架做改动后，应用于已知一个角度图片，生成另一个角度图片的任务。首先生成一个模糊的框架，这里用到了变分的方法。之后将模糊框架和原图一起通过 Conditional GAN 生成另一个视角的图片³¹，具体网络架构使用了 UNet。

5.3 Adversarial Training For Sketch Retrieval[9]

这篇应用文章使用 GAN 的角度很奇特，他使用了 GAN 来提取 feature。普通的卷积网络可以提取 feature，但监督学习的过程必须有 label。GAN 的 D 最后一层也可以认为是提取 feature，但是是无监督的。

作者图像检索 Merchant Marks，问题大概是这样的。测试了两种普通的卷积神经网络

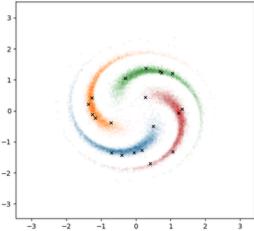


Figure 1: Labeled and unlabeled data are denoted by cross and point respectively, and different colors indicate classes.

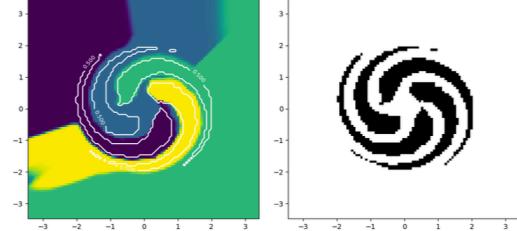
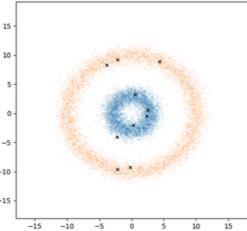


Figure 2: Left: Classification decision boundary, where the white line indicates true-fake boundary; Right: True-Fake decision boundary

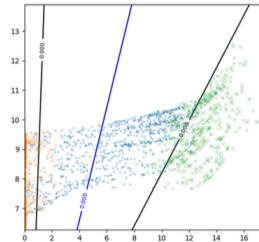


Figure 3: Feature space at convergence

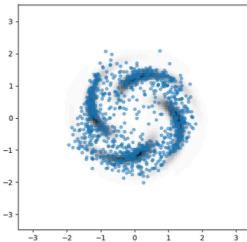


Figure 4: Left: Blue points are generated data, and the black shadow indicates unlabeled data. Middle and right can be interpreted as above.

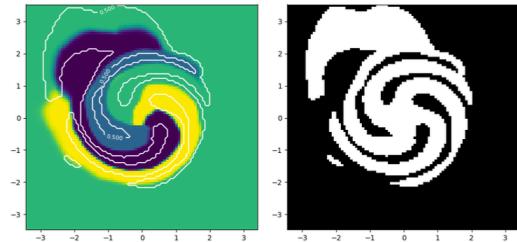


Figure 4: Left: Blue points are generated data, and the black shadow indicates unlabeled data. Middle and right can be interpreted as above.

Figure 30: Semi-supervised GAN 图解

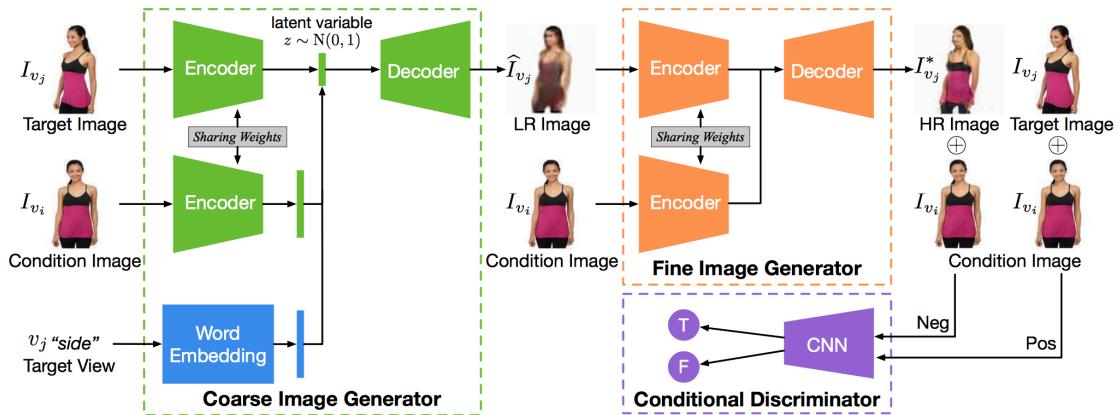


Figure 2. Architecture of the proposed VariGAN. It consists of three modules: coarse image generator, fine image generator and conditional discriminator. During training, a LR Image is firstly generated by the coarse image generator conditioned on the target image, conditioned image and target view. The fine image generator with skip connections is designed to generate the HR image. Finally, the HR image and the conditioned image are concatenated as negative pairs and passed to the conditional discriminator together with positive pairs (target image & condition image) to distinguish real and fake.

Figure 31: VariGAN 架构

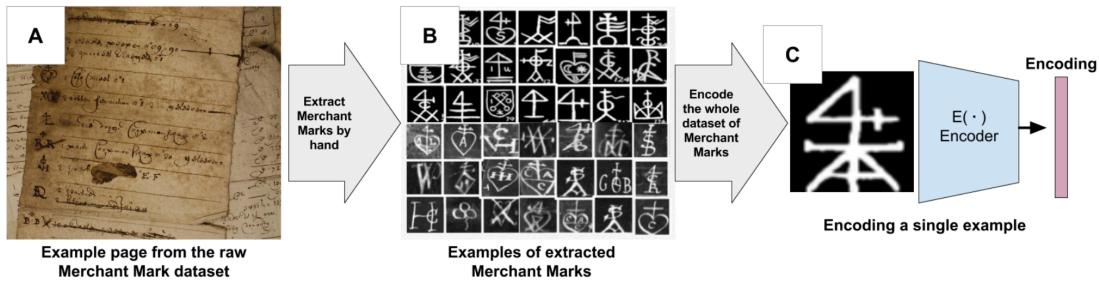


Figure 32: Sketch Retrieval 问题简介



Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Figure 33: SRGAN 与其他方法

框架，目测效果还不错。但是很奇怪没有定量地评价，也没有与其他提取 feature 的方法例如 autoencoder 等进行对比。

5.4 Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network[21]

使用 GAN 提高图像的分辨率，就是用 conditional GAN 生成高分辨率图像。loss 分为 3 部分，一部分是跟原始图像的 MSE，另一部分是 GAN 的 loss，还有一个 total variation 正则项。G 和 D 都是层数不太多的 ResNet。PS：PSNR 和 SSIM 都是越大越好。这里谈的是第五版的论文，本次更新发现直接使用深度残差网络的 SRResNet 取得了最好的效果，但是作者认为目前这两种定量指标都不足以说明人类的感知评价，于是使用了主观评价指标 MOS。确实感觉 SRGAN 得到的图片边缘比较锐利，人的主观评价得分比较高。

5.5 SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient[36]

问题 自然语言处理的序列生成的时候，使用 GAN 有两个问题：

- 离散样本问题，梯度没办法从 D 有效地传到 G。

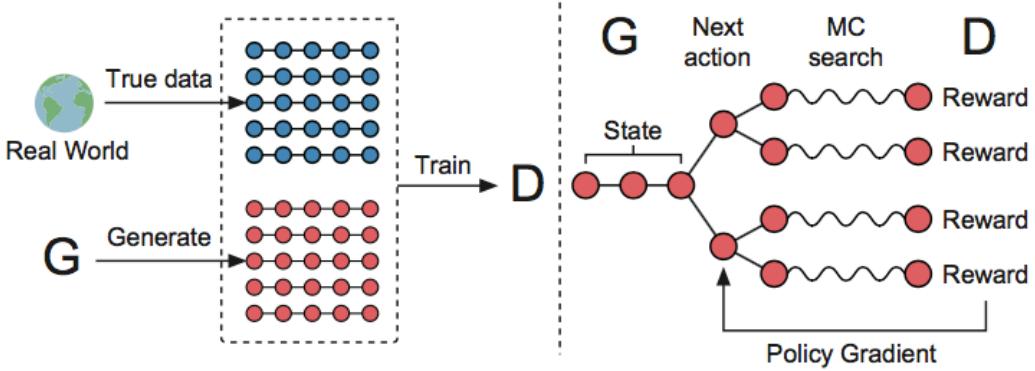


Figure 34: SeqGAN 图解

- GAN 只能对整个序列给定 loss，不能对中间结果进行评价，使得训练的针对性不强。

方法 解决问题的方法是用增强学习的 policy gradient 代替从 D 直接向 G 传的 gredient。定义 $G_\theta(y_t|Y_{1:t-1})$ 为对于前面序列，生成下面一个元素的模型或 policy。设 J 为生成模型的 reward，那么：

$$\nabla J(\theta) = \sum_{y \in \mathcal{Y}} (\nabla G_\theta(y_t|Y_{1:t-1})) Q_{D_\phi}^{G_\beta}(Y_{1:t-1}, y_t)$$

其中 Y 为某个 G_θ 生成的样本，Q 为回报估计函数：

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

也就是说，对于每一个自己生成的序列样本，只要 D 足够优秀，就可以计算生成模型在每一步生成元素的时候对应条件概率的梯度。这样一下子解决了上述两个问题：

- 离散样本没办法传梯度——原来 loss function 梯度的来源是 $\log(1 - D(G(z, \theta)))$ 需要通过样本传，现在强化学习直接定义期望回报为 $\sum_{y_1} G_\theta(y_1|s_0) \cdot Q_{D_\phi}^{G_\beta}(s_0, y_1)$ 是通过前面那部分传梯度的。
- 只能衡量完整样本——但我每一步都通过蒙特卡洛补成完整样本再算 reward

(4) 是上边的梯度公式，(5) 是 GAN 的交叉熵，(8) 是梯度下降。

5.6 Style Transfer Generative Adversarial Networks: Learning to Play Chess Differently[8]

这里的“风格”指的不是图像的风格，而是“棋风”，在某些情况下做出类似于某个人的下法。

毕竟是一篇应用的文章，想法很简单，就是 conditional GAN，输入棋盘状态，输出下一步，然后判别是不是特定棋风的下法。[36](#)

但是由于真实样本不够，在某些情况出现的时候必须得到一个不错的下法，因此 G 加入另一个 loss 项，用 log 似然保证预测的下法尽量接近大师棋谱中的下法、远离随机下法。

Algorithm 1 Sequence Generative Adversarial Nets

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

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

Figure 35: SeqGAN 算法描述

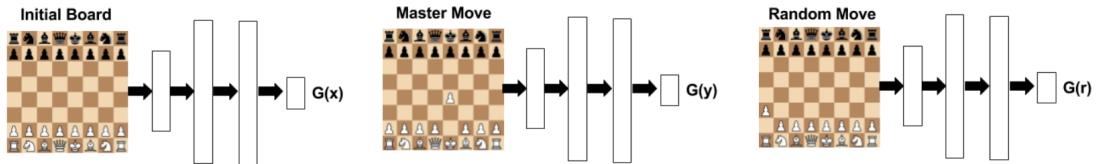


Figure 1: The generator network learns $G(x_G) = G(y_G)$, as well as $G(y_G) > G(r_G)$ if y_G corresponds to a move made by white and $G(y_G) < G(r_G)$ otherwise.

Figure 36: 棋风变换

5.7 SEGAN: Speech Enhancement Generative Adversarial Network[28]

这篇文章将 GAN 用于音频处理上，通过生成网络 G 除去噪音。对于信号的处理使用了一个 encoder-decoder 的结构，如³⁷。

训练使用了 Conditional GAN 的方法，如³⁸。

6 强化学习

6.1 Playing Atari with Deep Reinforcement Learning[24]

Question 比起监督学习，强化学习有以下几个问题：

1. 没有很多的数据，只有一个 sparse, noisy and delayed 的环境。感觉环境提供的信息其实比数据更多，只是不容易处理。delay 是最大的区别。2. 监督学习要求样本独立，强化学习前后状态有明显相关性。如果将整个决策、状态序列当成一个样本，就是独立的。但是状态空间太大，需要使用马尔可夫过程 MDP 的假设，只与前面的有关。3. 监督学习样本分布不变，而强化学习会随着你 policy 的更新而改变。这个应该是针对 Q-learning 等一类算法的，这里估计的长期 reward 依赖于某个 policy，而这个 reward 会被拿来做学习的 ground truth。

Background optimal action-value function

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$$

其中 R_t 为未来折扣回报 $R_t = \sum \gamma^{t'-t} r_{t'}$ Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s' \sim \varepsilon}[r_{t+1} + \gamma \max_{a'} Q^*(s', a') | s, a]$$

于是在把 s 看做序列的情况下可以计算，但状态量恐怖。

使用机器学习的思想，确定 $F(s)$ ，如果状态空间很大，不同状态的取值有一定相关性，那么参数化为 $f(s, \theta)$

于是

$$L(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)}(y_i - Q(s, a, \theta_i)), y_i = \mathbb{E}_{s' \sim \varepsilon}[r_{t+1} + \gamma \max_{a'} Q(s', a') | s, a]$$

Method 这里改进的主要采样方法使用了 ε -greedy 和每次随机更新之前采取过的 transition。这样相当于一个 smooth 的过程，才可能收敛。

Q 的输入有 $\phi(s)$ a ，二者很难直接并列输入，最后采取了根据不同 a 设置不同的最后一层的方法。

可能是由于决策的局部性，如果更新当前的 transition 的话容易被某种连续的策略 dominate。

6.2 Human-level control through deep reinforcement learning[25]

这是上一篇文章之后 DeepMind 发表在 Nature 上的一个改进版本。最重要的改进是把原来计算 ground truth 时的使用 $Q(\phi_{j+1}, a'; \theta)$ 改为一个参数更新有延迟版本的网络，更好地解决原来那个训练不稳定的问题。另外做了详细的测试⁴⁰。

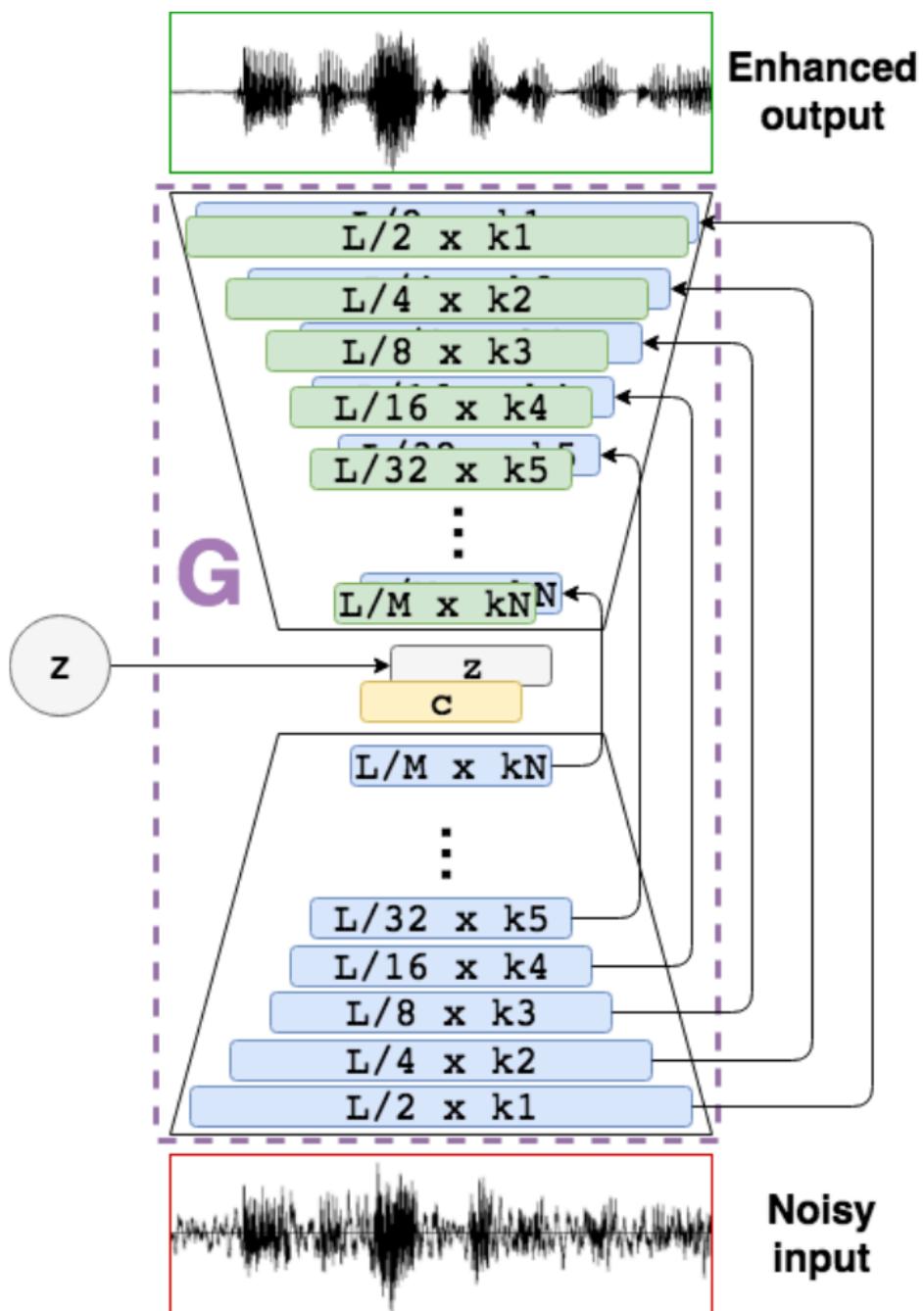


Figure 37: SEGAN 生成网络结构

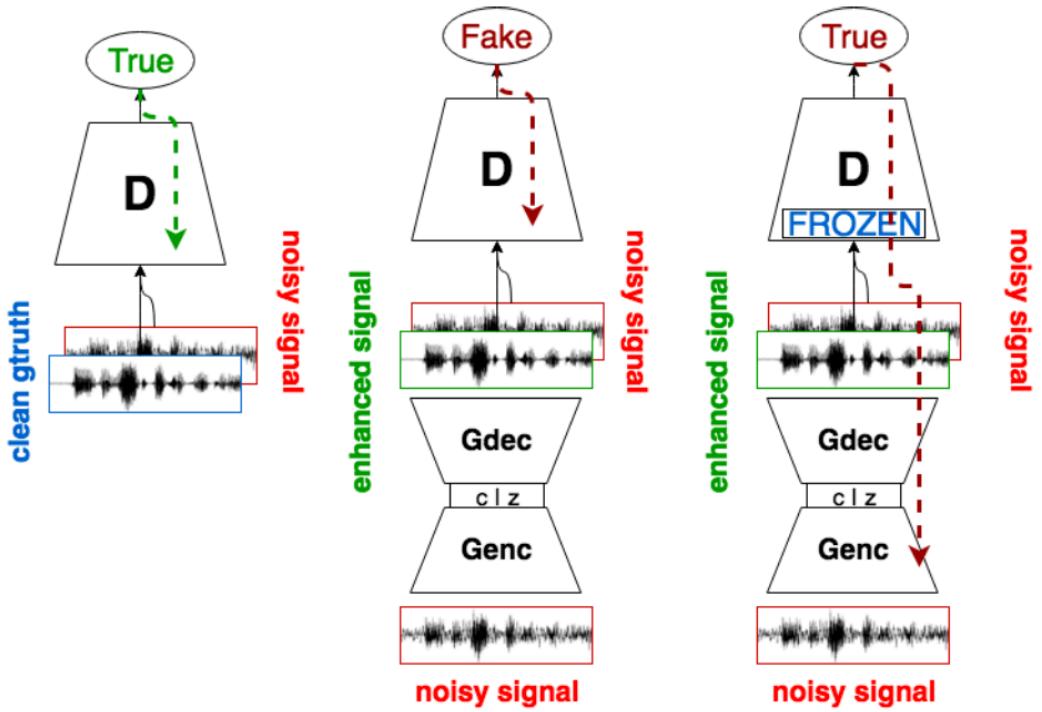


Figure 38: SEGAN 整体网络结构

Algorithm 1 Deep Q-learning with Experience Replay

```

Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
    Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
    for  $t = 1, T$  do
        With probability  $\epsilon$  select a random action  $a_t$ 
        otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
        Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
        Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
        Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
        Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
    end for
end for

```

Figure 39: DQN 算法描述

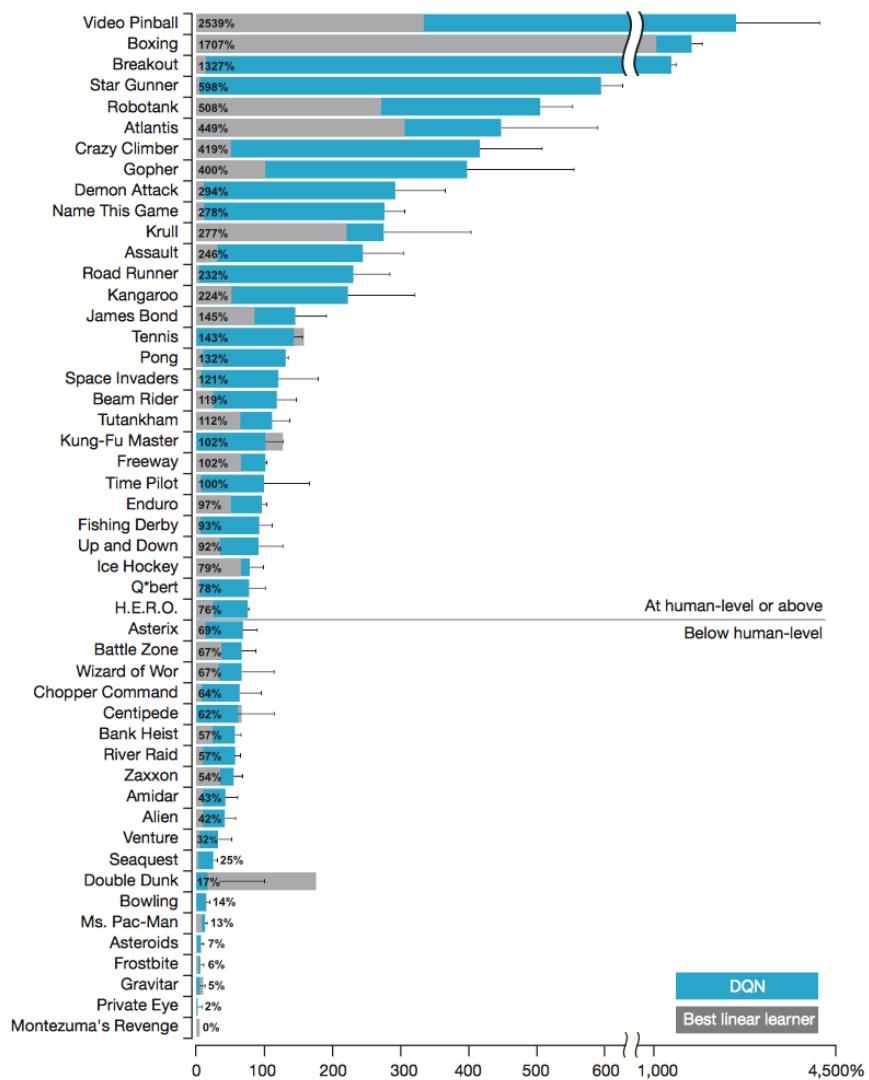


Figure 40: DQN 游戏水平

► **Double DQN:** Remove upward bias caused by $\max_a Q(s, a, \mathbf{w})$

- Current Q-network \mathbf{w} is used to **select** actions
- Older Q-network \mathbf{w}^- is used to **evaluate** actions

$$l = \left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}), \mathbf{w}^- \right) - Q(s, a, \mathbf{w})$$

<http://blog.csdn.net/u014210>

Figure 41: Double-DQN

6.3 Deep Reinforcement Learning with Double Q-learning[34]

针对目标函数 y 是 \max 得到的对结果有偏这个问题做出改进。首先 m estimates x , $\sum x - x_{real} = 0$, $\sum(x - x_{real})^2 = mC$ 可以证明

$$\max x_i \geq x_{real} + \sqrt{\frac{C}{m-1}}$$

证明很容易, 首先分成偏差为正负两类, 每类和一定, 使得方差最大显然要一个为和, 其他为 0。但是正的最大不得超过 $\sqrt{\frac{C}{m-1}}$ (反证), 所以最后取最优就在 $m-1$ 个正的 $\sqrt{\frac{C}{m-1}} - (m-1)\sqrt{\frac{C}{m-1}}$

(感觉不太符合题意, 毕竟并不是每个 action 都是最优 action 的无偏估计, 但是道理是这样的)

如何解决, 将 Nature DQN target network 选取最大的是按照 select network 选取的.....反正效果不错。

6.4 Dueling Network Architectures for Deep Reinforcement Learning[35]

Question 之前处理 s 和 action 输入关系的时候, 对每个 action 确定一个输出层, 这样不自然。

Method 将输入图像经过卷积处理之后, 分成两部分, 一部分计算出当前这个状态的价值, 另一部分计算执行 action 的价值。

$$Q(s, a; \theta, \alpha, \beta) = V(s, \theta, \alpha) + A(s, a, \beta)$$

注意要归一, 否则两部分相对规模不能确定。

$$Q(s, a; \theta, \alpha, \beta) = V(s, \theta, \alpha) + (A(s, a, \beta) - \max A(s, a, \beta))$$

减去最大值也可换成平均值。

References

- [1] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *CoRR*, abs/1701.04862, 2017. URL <http://arxiv.org/abs/1701.04862>.
- [2] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. URL <http://arxiv.org/abs/1701.07875>.

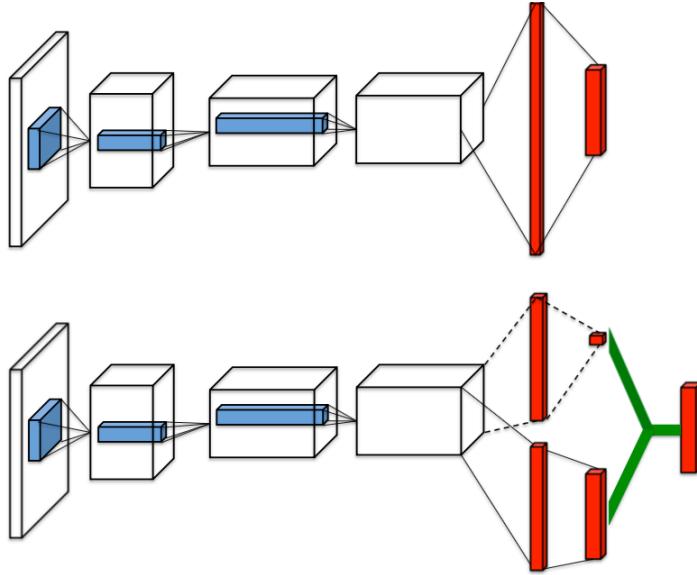


Figure 1. A popular single stream Q -network (**top**) and the dueling Q -network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q -values for each action.

Figure 42: Dueling DQN 图解

- [3] Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study. *CoRR*, abs/1706.08224, 2017. URL <http://arxiv.org/abs/1706.08224>.
- [4] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. *CoRR*, abs/1706.00826, 2017. URL <http://arxiv.org/abs/1706.00826>.
- [5] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717, 2017. URL <http://arxiv.org/abs/1703.10717>.
- [6] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. *arXiv preprint arXiv:1707.09405*, 2017.
- [7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016. URL <http://arxiv.org/abs/1606.03657>.
- [8] Muthuraman Chidambaram and Yanjun Qi. Style transfer generative adversarial networks: Learning to play chess differently. *CoRR*, abs/1702.06762, 2017. URL <http://arxiv.org/abs/1702.06762>.
- [9] Antonia Creswell and Anil Anthony Bharath. Adversarial training for sketch retrieval. *CoRR*, abs/1607.02748, 2016. URL <http://arxiv.org/abs/1607.02748>.

- [10] Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad GAN. *CoRR*, abs/1705.09783, 2017. URL <http://arxiv.org/abs/1705.09783>.
- [11] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015. URL <http://arxiv.org/abs/1506.05751>.
- [12] Carl Doersch. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908, 2016. URL <http://arxiv.org/abs/1606.05908>.
- [13] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL <http://arxiv.org/abs/1701.00160>.
- [14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL <http://arxiv.org/abs/1406.2661>.
- [15] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL <http://arxiv.org/abs/1308.0850>.
- [16] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. URL <http://arxiv.org/abs/1704.00028>.
- [17] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. *CoRR*, abs/1602.05110, 2016. URL <https://arxiv.org/abs/1602.05110>.
- [18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL <http://arxiv.org/abs/1611.07004>.
- [19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL <http://arxiv.org/abs/1312.6114>.
- [20] Hanock Kwak and Byoung-Tak Zhang. Generating images part by part with composite generative adversarial networks. *CoRR*, abs/1607.05387, 2016. URL <http://arxiv.org/abs/1607.05387>.
- [21] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016. URL <http://arxiv.org/abs/1609.04802>.
- [22] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. *CoRR*, abs/1703.02291, 2017. URL <http://arxiv.org/abs/1703.02291>.
- [23] Xiaodan Liang, Hao Zhang, and Eric P. Xing. Generative semantic manipulation with contrasting gan. *CoRR*, abs/1708.00315, 2017. URL <https://arxiv.org/abs/1708.00315>.

- [24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL <http://arxiv.org/abs/1312.5602>.
- [25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- [26] Augustus Odena. Semi-supervised learning with generative adversarial networks. *CoRR*, abs/1606.01583, 2016. URL <http://arxiv.org/abs/1606.01583>.
- [27] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *CoRR*, abs/1610.09585, 2016. URL <http://arxiv.org/abs/1610.09585>.
- [28] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. SEGAN: speech enhancement generative adversarial network. *CoRR*, abs/1703.09452, 2017. URL <http://arxiv.org/abs/1703.09452>.
- [29] Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *CoRR*, abs/1701.06264, 2017. URL <http://arxiv.org/abs/1701.06264>.
- [30] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL <http://arxiv.org/abs/1511.06434>.
- [31] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL <http://arxiv.org/abs/1606.03498>.
- [32] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017. URL <http://arxiv.org/abs/1701.05517>.
- [33] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *CoRR*, abs/1611.02200, 2016. URL <http://arxiv.org/abs/1611.02200>.
- [34] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL <http://arxiv.org/abs/1509.06461>.
- [35] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015. URL <http://arxiv.org/abs/1511.06581>.
- [36] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473, 2016. URL <http://arxiv.org/abs/1609.05473>.

- [37] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016. URL <http://arxiv.org/abs/1612.03242>.
- [38] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, and Jiashi Feng. Multi-view image generation from a single-view. *CoRR*, abs/1704.04886, 2017. URL <http://arxiv.org/abs/1704.04886>.
- [39] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016. URL <http://arxiv.org/abs/1609.03126>.
- [40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. URL <http://arxiv.org/abs/1703.10593>.