VIII. Generative Adversarial Networks

Young-geun Kim

Department of Statistics and Probability

STT 997 (SS 2025)

MICHIGAN STATE
UNIVERSITY

Recap of Previous Section

- Most previously reviewed methods, such as independent component analysis (ICA), auto-regressive models, restricted Boltzmann machines (RBMs), and variational autoencoders (VAEs), explicitly formulate density models.
- However, the intractability of likelihoods due to non-linear generators often limits their model classes to those with tractable density forms (e.g., (nonlinear) ICA, auto-regressive models).
- Additionally, this limitation forces the use of surrogate density forms (e.g., multi-step Gibbs sampling in RBMs, variational inference in VAEs).

Recap of Previous Section

- For example, in RBMs, the training with contrastive divergence algorithm is based on the following formulation:

$$\partial \log p_\theta(\vec{x})/\partial\theta = -\sum_{\vec{z}} \Big(\partial E_\theta(\vec{z},\vec{x})/\partial\theta\Big) p_\theta(\vec{z}|\vec{x}) + \sum_{\vec{z},\vec{x}} \Big(\partial E_\theta(\vec{z},\vec{x})/\partial\theta\Big) p_\theta(\vec{z},\vec{x}) \quad (1)$$

  where each term is approximated using a Markov chain.

- As another example, in VAEs, the likelihood is approximated with the evidence lower bound (ELBO):

$$\log p_\theta(\vec{x}) - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z}|\vec{x})) \quad (2)$$

  where the approximation becomes accurate when we have perfect inference networks, $q_\phi(\vec{z}|\vec{x}) \approx p_\theta(\vec{z}|\vec{x})$.

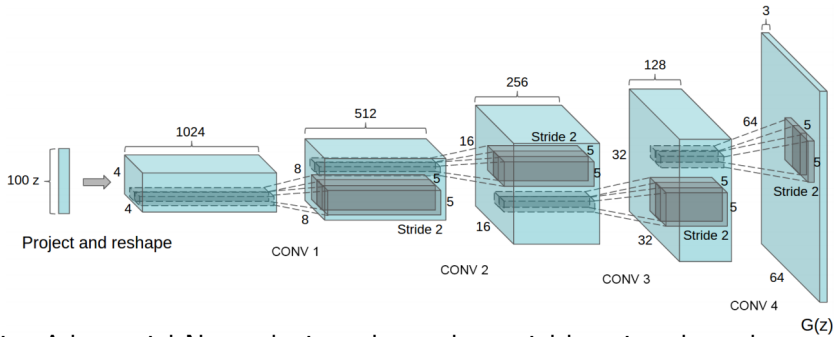
Recap of Previous Section

- Despite their formality, approximations of (gradients of) log-likelihoods in RBMs and VAEs often fail, resulting in blurry generation results.
- Generative Adversarial Networks (GANs; Goodfellow et al., 2014) propose an alternative approach to approximating a likelihood ratio-based objective by leveraging a discriminative model, without explicit density models.

Images are from Tolstikhin et al. (2018).

Outline

Generative Adversarial Network: Generator



- Generative Adversarial Networks introduce adversarial learning through two networks: (i) a generator and (ii) a discriminator.
- The generator specifies the same graphical model used in VAEs, but $\vec{x} = G_\theta(\vec{z})$ where G is a neural network, i.e., $p_\theta(\vec{x}|\vec{z})$ degenerates to a single point.

Example of the generator architecture of deep convolutional GANs (Radford et al., 2016).

Generative Adversarial Network: Generator

- The distribution of generated data can be expressed as:

$$\begin{aligned}\mathbb{P}_\theta(\vec{x}) &:= \int \mathbb{P}_\theta(\vec{x}|\vec{z})p(\vec{z})d\vec{z} \\ &= \int \delta_{\vec{x}}(G_\theta(\vec{z}))p(\vec{z})d\vec{z}.\end{aligned} \tag{3}$$

  For a given $\vec{x}$ and θ, to calculate this quantity, we need to find all $\vec{z}$ such that $G_\theta(\vec{z}) = \vec{x}$ and sum their densities.
- (vs. Variational Autoencoder) $p_\theta(\vec{x}|\vec{z}) = p(\vec{x}; \vec{\mu}_{\vec{X}|\vec{Z}}(\vec{z}), \Sigma_{\vec{X}|\vec{Z}}(\vec{z}))$.
- (vs. (Nonlinear) Independent Component Analysis) When the generator is invertible, and its inverse has a tractable form, we can compute likelihoods via the change of variables formula:

$$p_\theta(\vec{x} = G_\theta(\vec{z})) = p(\vec{z} = G_\theta^{-1}(\vec{x})) \left| \frac{\partial G_\theta^{-1}(\vec{x})}{\partial \vec{x}} \right|. \tag{4}$$

Generative Adversarial Network: Adversarial Learning

- Again, GANs introduce a discriminator. The key idea is to train a binary classifier that distinguishes real data from generated ones, and to update the generator ($G_\theta(\vec{z})$) to fool it.
- The discriminator is trained to classify data from the following two classes:

$$(\text{Real Data}; Y = 1)\ \vec{x}_1, \ldots, \vec{x}_n \sim p(\vec{x}) \tag{5}$$

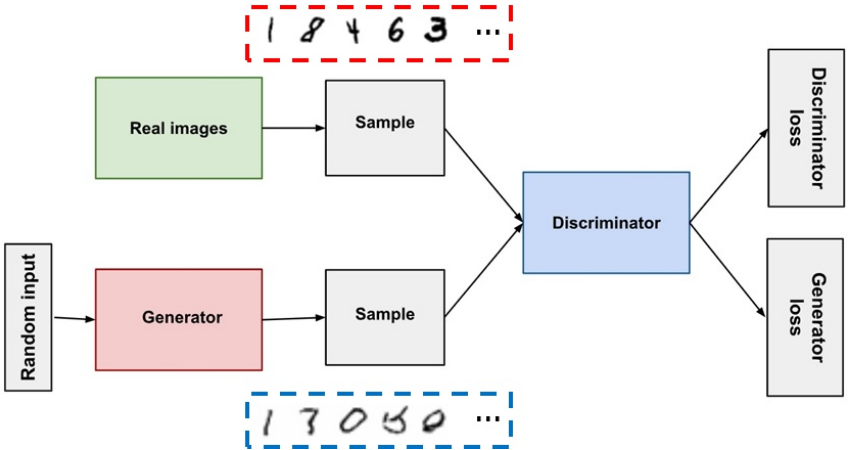
and

$$(\text{Generated Data}; Y = 0)\ G_\theta(\vec{z}_1), \ldots, G_\theta(\vec{z}_n),\ \text{where } \vec{z}_1, \ldots, \vec{z}_n \sim p(\vec{z}). \tag{6}$$

We denote the output of the discriminator as $D_\phi(\vec{x})$, which models $\mathbb{P}(Y = 1 | \vec{X} = \vec{x})$.

- The generator G_θ is then updated to maximize the loss of the discriminator D_ϕ. This process is iteratively repeated, which is referred to as *adversarial learning*.

Generative Adversarial Network: Adversarial Learning



Images were edited from https://developers.google.com/machine-learning/gan/gan_structure and https://github.com/MorvanZhou/mnistGANs.

Generative Adversarial Network: Adversarial Learning

- Adversarial learning can be formulated as follows.
- First, the loss function of the discriminator D_ϕ is the cross-entropy loss:

$$l(D_\phi(\vec{x}), y) = -I(y = 1) \log D_\phi(\vec{x}) - I(y = 0) \log \Big(1 - D_\phi(\vec{x})\Big). \tag{7}$$

- With the dataset consisting of n real data and n generated data, $(\vec{x}_1, y_1 = 1), \ldots, (\vec{x}_n, y_n = 1), (G_\theta(\vec{z}_1), y_{n+1} = 0), \ldots, (G_\theta(\vec{z}_n), y_{2n} = 0)$, the empirical risk can be expressed as:

$$-n^{-1} \sum_{i=1}^{n} \log D_\phi(\vec{x}_i) - n^{-1} \sum_{i=1}^{n} \log(1 - D_\phi(G_\theta(\vec{z}_i))). \tag{8}$$

- Next, the loss function of the generator G_θ is $-l(D_\phi(G_\theta(\vec{z})), y = 0)$, and the objective function can be expressed as:

$$n^{-1} \sum_{i=1}^{n} \log(1 - D_\phi(G_\theta(\vec{z}_i))). \tag{9}$$

Generative Adversarial Network: Adversarial Learning

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

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

  end for

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

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

end for

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

The pseudo-algorithm is from Goodfellow et al. (2014).

Generative Adversarial Network: Adversarial Learning

- The adversarial learning process involves alternately maximizing and minimizing the negative cross-entropy loss:

$$V(\theta, \phi) := \int \Big(\log D_\phi(\vec{x}) \Big) p_n(\vec{x}) \, d\vec{x} + \int \Big(\log(1 - D_\phi(\vec{x})) \Big) p_\theta(\vec{x}) \, d\vec{x}. \tag{10}$$

- This process can be viewed as a two-player minimax game where the goal is to find

$$\theta^* \in \arg\min_\theta \Big(\max_\phi V(\theta, \phi) \Big). \tag{11}$$

- The adversarial learning can be interpreted as repeated cycles of approximating and minimizing the Jensen-Shannon (JS) divergence:

$$\max_\phi V(\theta, \phi) = \mathrm{JS}(p_n \parallel p_\theta) := \frac{1}{2} \left(\mathrm{KL} \left(p_n || \frac{p_n + p_\theta}{2} \right) + \mathrm{KL} \left(p_\theta || \frac{p_n + p_\theta}{2} \right) \right). \tag{12}$$

up to a constant addition and sign-preserving multiplication. Thus, p_{θ^*} is the minimizer of $\mathrm{JS}(p_n \parallel p_\theta)$.

Generative Adversarial Network: Adversarial Learning

- We assume that the discriminator class $\{D_\phi \mid \phi \in \Phi\}$ is sufficiently flexible such that for any given θ, there exists a $\phi^*(\theta)$ where:

$$D_{\phi^*(\theta)}(\vec{x}) = \frac{p_n(\vec{x})}{p_n(\vec{x}) + p_\theta(\vec{x})}. \tag{13}$$

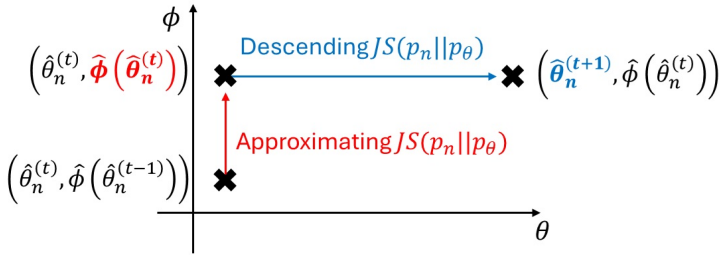
Proof: $V(\theta, \phi) = \int \Big(\log D_\phi(\vec{x}) p_n(\vec{x}) + \log(1 - D_\phi(\vec{x})) p_\theta(\vec{x}) \Big) d\vec{x}$, and the integrand is strictly concave w.r.t. $D_\phi(\vec{x})$. The first derivative of the integrand w.r.t. $D_\phi(\vec{x})$ is

$$-\frac{p_n(\vec{x}) + p_\theta(\vec{x})}{D_\phi(\vec{x})(1 - D_\phi(\vec{x}))} \left(D_\phi(\vec{x}) - \frac{p_n(\vec{x})}{p_n(\vec{x}) + p_\theta(\vec{x})} \right), \tag{14}$$

implying that $V(\theta, \phi)$ is maximized when Equation (13) holds. This implies

$$\max_\phi V(\theta, \phi) = V(\theta, \phi^*(\theta)) = 2\mathrm{JS}(p_n \parallel p_\theta) - \log 4. \tag{15}$$

Generative Adversarial Network: Adversarial Learning



- The two adversarial networks, the discriminator and the generator, are trained alternately:

  1. Given $\hat{\theta}_n^{(t)}$, update the discriminator to obtain $\hat{\phi}(\hat{\theta}_n^{(t)})$ by maximizing $V(\hat{\theta}_n^{(t)}, \phi)$.
  2. Given $\hat{\phi}(\hat{\theta}_n^{(t)})$, update the generator to obtain $\hat{\theta}_n^{(t+1)}$ by minimizing $V(\theta, \hat{\phi}(\hat{\theta}_n^{(t)}))$.
  3. Repeat the above processes.

Generative Adversarial Network: Generation Result

Images are from Goodfellow et al. (2014).

Generative Adversarial Network: Inference

- In (vanilla) GANs, there is no density-based inference, as they do not have explicit density models.
- Several variants of GANs have been proposed to learn (interpretable) representations and conduct corresponding inferences. They will be discussed in the next subsection for *representation learning with adversarial learning*.

Outline

Information Maximizing Generative Adversarial Network

- Information Maximizing GANs (InfoGANs) (Chen et al., 2016) decompose the latent factors into two parts: (i) signal-related factors ($\vec{C}$) and (ii) noise-related factors ($\vec{Z}$).
- They modify the generator to $G_\theta(\vec{Z}, \vec{C})$ and the value function as:

$$V(\theta, \phi) - \lambda \, \mathrm{MI}(\vec{C}; G_\theta(\vec{Z}, \vec{C})) \tag{16}$$

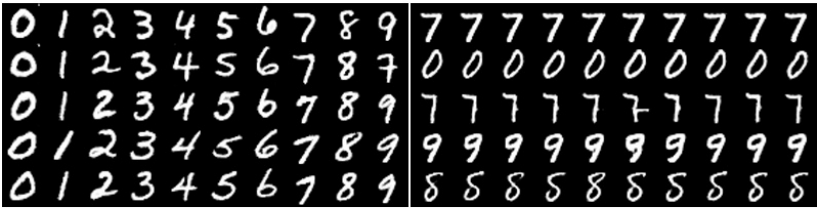
where $\mathrm{MI}(\vec{C}; G_\theta(\vec{Z}, \vec{C})) := \mathrm{KL}(p(\vec{c})p_\theta(G_\theta(\vec{z}, \vec{c})|\vec{c})||p(\vec{c})p(G_\theta(\vec{z}, \vec{c})))$ represents the mutual information between the signal-related factor $\vec{C}$ and the generated data $G_\theta(\vec{Z}, \vec{C})$, and λ is a hyperparameter.

- The new penalty term is constant w.r.t. ϕ, so the maximization over ϕ yields:

$$(2 \, \mathrm{JS}(p_n||p_\theta) - \log 4) - \lambda \, \mathrm{MI}(\vec{C}; G_\theta(\vec{Z}, \vec{C})). \tag{17}$$

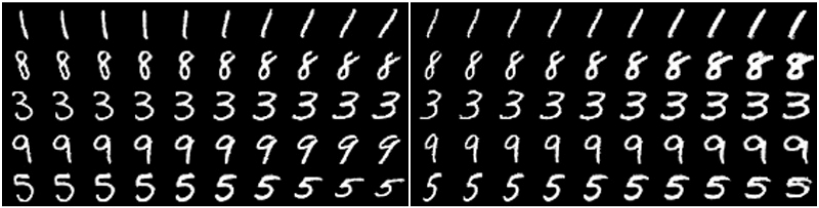
The penalty encourages $\vec{C}$ to be dependent on $G_\theta(\vec{Z}, \vec{C})$.

InfoGAN: Generation Result



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



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

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

Images are from Chen et al. (2016).

Bidirectional Generative Adversarial Network

- Bidirectional GANs (BiGANs) (Donahue et al., 2016) introduce an encoder $E_\psi(\vec{x})$ to directly learn the dimension-reduction mapping.[1]
- The discriminator is modified to classify pairs $(\vec{X}, E_\psi(\vec{X}))$ vs. $(G_\theta(\vec{Z}), \vec{Z})$. The corresponding value function can be expressed as:

$$V(\theta, \phi, \psi) := \int \Big(\log D_\phi(\vec{x}, E_\psi(\vec{x})) \Big) p_n(\vec{x})\, d\vec{x} + \int \Big(\log(1 - D_\phi(G_\theta(\vec{z}), \vec{z})) \Big) p(\vec{z})\, d\vec{z}. \quad (18)$$

- The maximized value function over ϕ is $\mathrm{JS}(p_n(\vec{x}, E_\psi(\vec{x}))||p_\theta(G_\theta(\vec{z}), \vec{z}))$, and is minimized w.r.t. encoder and generator parameters (θ, ψ).
- The optimal encoder and generator identify an ideal autoencoder that perfectly reconstruct $\vec{x}$.

[1] Adversarially Learned Inference (Dumoulin et al., 2016) is a concurrently proposed work with a similar idea.

Bidirectional Generative Adversarial Network: Architecture

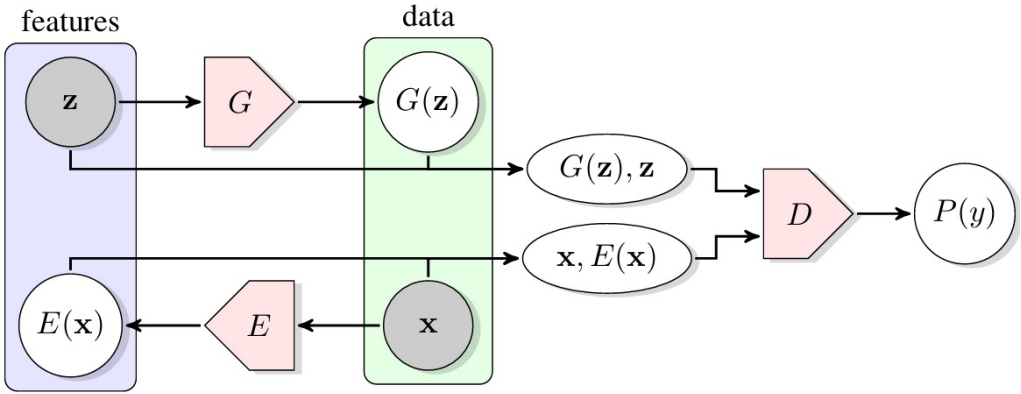
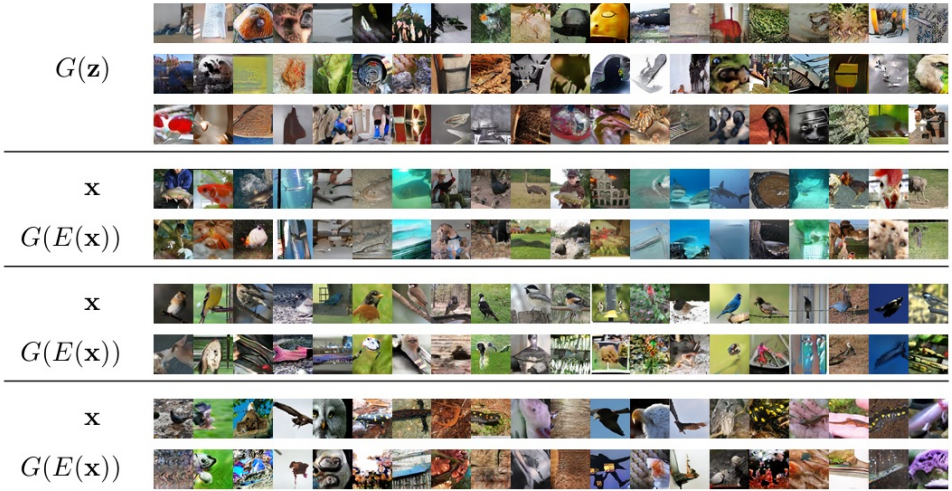


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

Images are from Donahue et al. (2016).

Bidirectional Generative Adversarial Network: Result



Images are from Donahue et al. (2016).

Adversarially Learned Anomaly Detection

- Despite the theory suggesting that the optimal encoder and generator perfectly reconstruct $\vec{x}$, it has been empirically observed that this may not hold (Li et al., 2017).
- Adversarially Learned Anomaly Detection (Zenati et al., 2018) introduces adversarial learning about the reconstructed results:

$$
\begin{aligned}
&V(\theta, \phi, \psi) \\
&:= \int \Big(\log D_{XZ,\phi}(\vec{x}, E_\psi(\vec{x})) \Big) p_n(\vec{x})\, d\vec{x} + \int \Big(\log(1 - D_{XZ,\phi}(G_\theta(\vec{z}), \vec{z})) \Big) p(\vec{z})\, d\vec{z} \\
&\quad + \int \Big(\log D_{XX,\phi}(\vec{x}, \vec{x}) \Big) p_n(\vec{x})\, d\vec{x} + \int \Big(\log(1 - D_{XX,\phi}(\vec{x}, G_\theta(E_\psi(\vec{x})))) \Big) p_n(\vec{x})\, d\vec{x} \\
&\quad + \int \Big(\log D_{ZZ,\phi}(\vec{z}, \vec{z}) \Big) p(\vec{z})\, d\vec{z} + \int \Big(\log(1 - D_{ZZ,\phi}(\vec{z}, E_\psi(G_\theta(\vec{z})))) \Big) p(\vec{z})\, d\vec{z}.
\end{aligned}
\tag{19}
$$

Here, $D_{XZ,\phi}$, $D_{XX,\phi}$, and $D_{ZZ,\phi}$ are all discriminators.

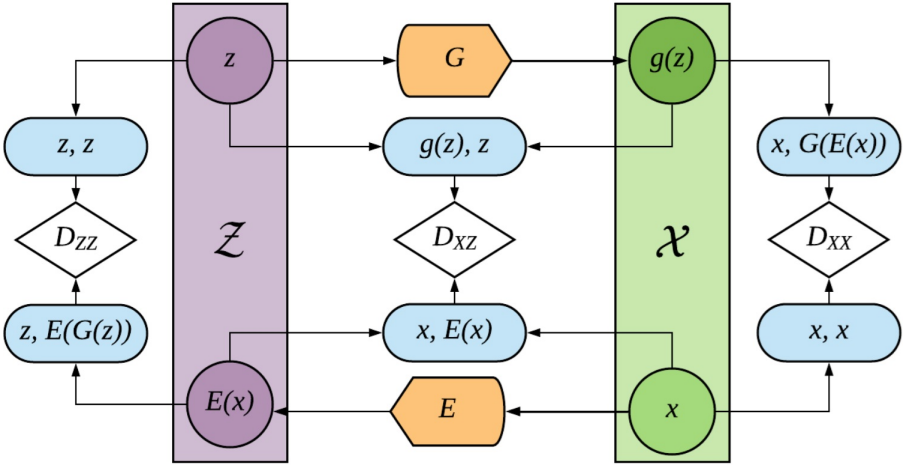
Adversarially Learned Anomaly Detection

- The maximized value function over the discriminator parameter ϕ is:

$$\begin{aligned}\max_{\phi} V(\theta, \phi, \psi) = &\Big(2\,\mathrm{JS}(p_n(\vec{x}, E_\psi(\vec{x}))||p_\theta(G_\theta(\vec{z}), \vec{z})) - \log 4\Big) \\ &+ \Big(2\,\mathrm{JS}(p_n(\vec{x}, \vec{x})||p_{\theta,\psi}(\vec{x}, G_\theta(E_\psi(\vec{x})))) - \log 4\Big) \\ &+ \Big(2\,\mathrm{JS}(p(\vec{z}, \vec{z})||p_{\theta,\psi}(\vec{z}, E_\psi(G_\theta(\vec{z})))) - \log 4\Big).\end{aligned} \tag{20}$$

- The optimal encoder and generator still identify an ideal autoencoder.

Adversarially Learned Anomaly Detection: Architecture



Images are from Zenati et al. (2018).

Adversarially Learned Anomaly Detection: Anomaly Score

Algorithm 1 Adversarially Learned Anomaly Detection

Input $\boldsymbol{x}, \sim p_{\mathcal{X}_{Test}}(x), E, G, f_{xx}$ where f_{xx} is the feature layer of D_{xx}

Output $A(\boldsymbol{x})$, where A is the anomaly score

1: **procedure** INFERENCE
2: $\tilde{\boldsymbol{z}} \leftarrow E(\boldsymbol{x})$ ▷ Encode samples
3: $\hat{\boldsymbol{x}} \leftarrow G(\tilde{\boldsymbol{z}})$, ▷ Reconstruct samples
4: $f_\delta \leftarrow f_{xx}(\boldsymbol{x}, \hat{\boldsymbol{x}})$
5: $f_\alpha \leftarrow f_{xx}(\boldsymbol{x}, \boldsymbol{x})$
6: return $\|f_\delta - f_\alpha\|_1$
7: **end procedure**

The algorithm is from Zenati et al. (2018).

Adversarially Learned Anomaly Detection: Result

Figure 5. Error analysis on SVHN when the normal class is the "3" digit. 1st row: normal data; 2nd row: reconstruction of normal data; 3rd row: anomalous data; 4th row: reconstruction of anomalous data.

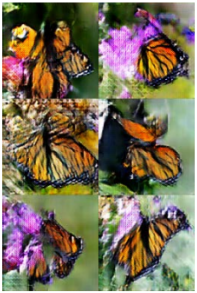
Images are from Zenati et al. (2018).

Outline

Unconditional vs. Conditional Generation

- Synthetic data generation in deep generative models is roughly divided by (i) unconditional and (ii) conditional approaches.
  1. Unconditional: $\vec{X} = G_\theta(\vec{Z})$ where the latent variables $\vec{Z}$ are sampled from known prior distribution.
  2. Conditional: $\vec{X} = G_\theta(\vec{Z}, \vec{C})$ (or simply $\vec{X} = G_\theta(\vec{C})$) where $\vec{C}$ are additional covariates (observed conditioning data).
- This subsection focuses on (ii) conditional approaches. Most of approaches are extended from methods for (i).
- According to the dimensionality of the conditioning data $\vec{C}$, the (ii) is further divided into class-conditional generation and controllable generation.

Class-Conditional Generation


monarch butterfly


goldfinch


daisy

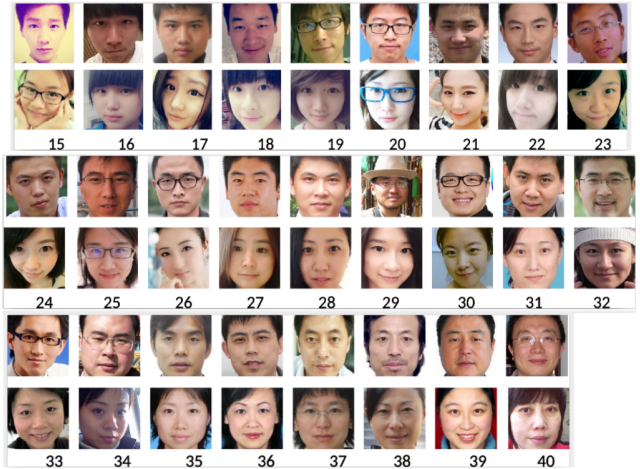

redshank


grey whale

- The conditioning data $\vec{C}$ is usually a class label, but continuous variables such as demographic attributes are also used frequently.

Conditional image synthesis results by a model trained on the ImageNet dataset (Odena et al., 2017). All images are generated by feed-forwarding random noise and class labels.

Class-Conditional Generation: Example Dataset



Face images in the asian face age dataset from various ages (Niu et al., 2016).

Class-Conditional Generation: Example Dataset

Figure: Face images in extended Yale B dataset from various light conditions. From left to right, the azimuth of light source changes from -110 to +110 in degree.

Figure: Face images in multi-pie dataset from various camera conditions. The azimuth of camera changes from -90 to 90 in degree.

Class-Conditional Generation: Adversarial Learning

- We denote the dataset with additional conditioning data by $\{(\vec{x}_i, \vec{c}_i)\}_{i=1}^n$. The generator G_θ inputs $\vec{C}$ in addition to $\vec{Z}$.
- The discriminator D_ϕ is modified to classify $(\vec{X}, \vec{C})$ $(Y = 1)$ vs. $(G_\theta(\vec{Z}, \vec{C}), \vec{C})$ $(Y = 0)$.
- The value function can be formulated as:

$$V(\theta, \phi) := \int \Big(\log D_\phi(\vec{x}, \vec{c}) \Big) p_n(\vec{x}, \vec{c})\, d\vec{x} d\vec{c} + \int \Big(\log(1 - D_\phi(G_\theta(\vec{z}, \vec{c}), \vec{c})) \Big) p(\vec{z}) p(\vec{c})\, d\vec{z} d\vec{c}. \tag{21}$$

Here, $p_n(\vec{x}, \vec{c})$ represents the empirical distribution of $(\vec{X}, \vec{C})$ and $p(\vec{z})p(\vec{c})$ indicates that $\vec{Z}$ and $\vec{C}$ are independent.

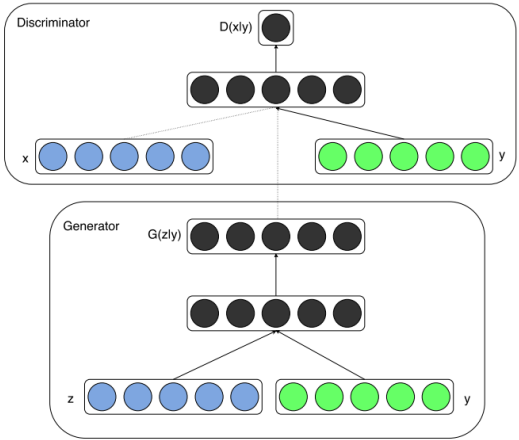
Class-Conditional Generation: Adversarial Learning

- The maximized value function over D_ϕ can be expressed as:

$$\max_{\phi} V(\theta, \phi) = \Big(2\,\mathrm{JS}(p_n(\vec{x}, \vec{c}) || p_\theta(\vec{x}, \vec{c})) - \log 4\Big). \tag{22}$$

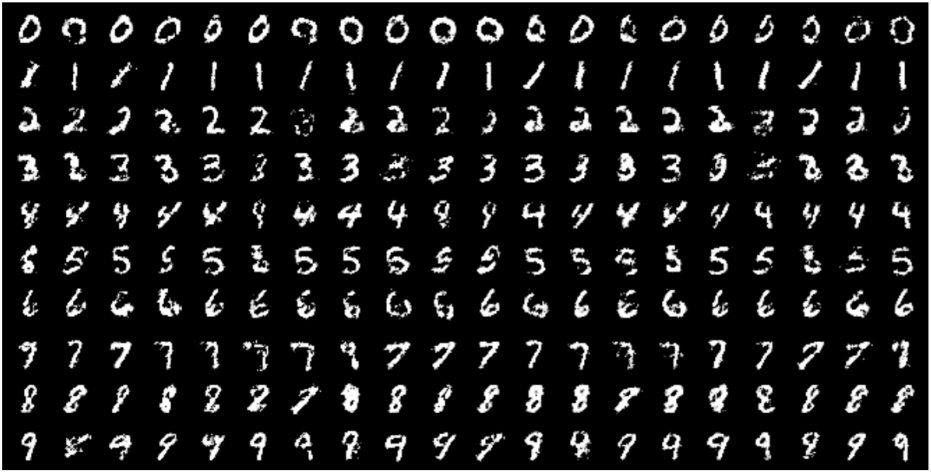
  where $p_\theta(\vec{z}, \vec{x}, \vec{c}) := p(\vec{z})p(\vec{c})p_\theta(\vec{x}|\vec{z}, \vec{c})$.
- That is, minimizing the maximized value function aims to learn a generator G_θ s.t. $\mathbb{P}_{(G_\theta(\vec{Z},\vec{C}),\vec{C})} = \mathbb{P}_{(\vec{X},\vec{C})}$. This is equivalent to $\mathbb{P}_{G_\theta(\vec{Z},\vec{C})|\vec{C}=\vec{c}} = \mathbb{P}_{\vec{X}|\vec{C}=\vec{c}}$ for every realization $\vec{c}$.

Class-Conditional Generation: Architecture



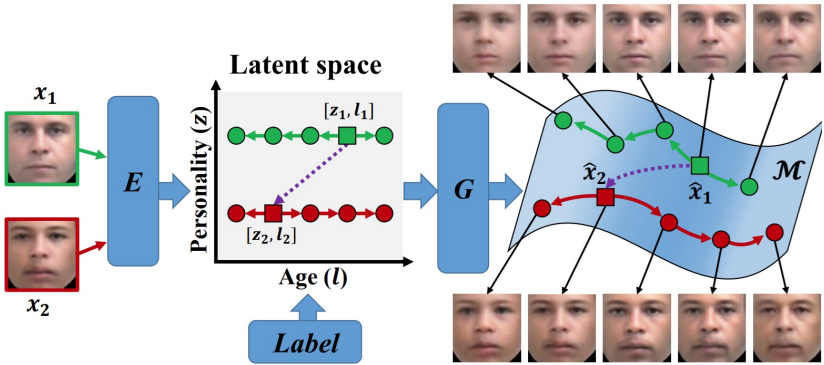
Architectures of discriminator and generator networks in conditional GANs (Mirza and Osindero, 2014). Both networks input class labels.

Class-Conditional Generation: Result



Images are from Mirza and Osindero (2014).

Class-Conditional Generation: Introducing Autoencoder



- Some works have introduced encoders.

The figure is from Zhang et al. (2017).

Class-Conditional Generation: Introducing Autoencoder

- For example, in Zhang et al. (2017), the value function can be expressed as:

$$
\begin{aligned}
V(\theta, \phi, \psi) := & \left(\int \Big(\log D_{XC,\phi}(\vec{x}, \vec{c}) \Big) p_n(\vec{x}, \vec{c})\, d\vec{x} d\vec{c} \right. \\
& \left. + \int \Big(\log(1 - D_{XC,\phi}(G_\theta(E_\psi(\vec{x}), \vec{c}), \vec{c})) \Big) p_n(\vec{x}, \vec{c})\, d\vec{x} d\vec{c} \right) \\
& + \left(\int \Big(\log D_{Z,\phi}(\vec{z}) \Big) p(\vec{z})\, d\vec{z} + \int \Big(\log(1 - D_{Z,\phi}(E_\psi(\vec{x}))) \Big) p_n(\vec{x})\, d\vec{x} \right).
\end{aligned} \tag{23}
$$

Class-Conditional Generation: Introducing Autoencoder

- The objective function is the summation of the value function and the reconstruction error:

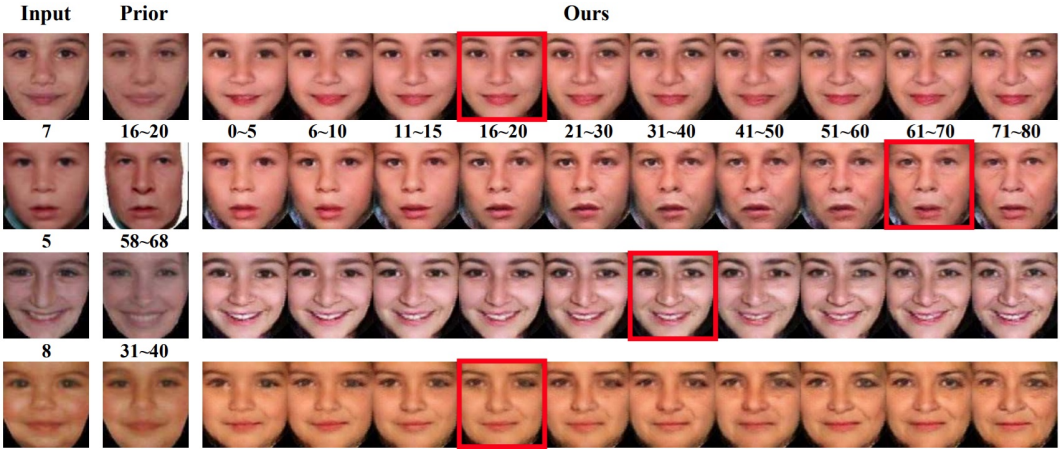
$$\lambda \int ||\vec{x} - G_\theta(E_\psi(\vec{x}), \vec{c})|| p_n(\vec{x}, \vec{c}) d\vec{x} d\vec{c} + V(\theta, \phi, \psi) \tag{24}$$

  where λ is a hyperparameter. The authors added a total variation-based penalty term to remove noisy artifacts in generated images.

- Note that the first term, the reconstruction error, does not depend on ϕ. After maximizing the objective w.r.t. ϕ, we have:

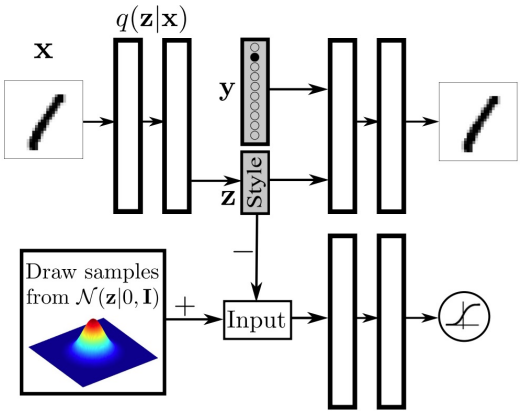
$$\begin{aligned} &\lambda \int ||\vec{x} - G_\theta(E_\psi(\vec{x}), \vec{c})|| p_n(\vec{x}, \vec{c}) d\vec{x} d\vec{c} \\ &+ \Big(2\,\mathrm{JS}(p_n(\vec{x}, \vec{c}) || p_{\theta,\psi}(G_\theta(E_\psi(\vec{x}), \vec{c}), \vec{c})) - \log 4\Big) \\ &+ \Big(2\,\mathrm{JS}(p(\vec{z}) || p_\psi(E_\psi(\vec{x}))) - \log 4\Big). \end{aligned} \tag{25}$$

Class-Conditional Generation: Introducing Autoencoder



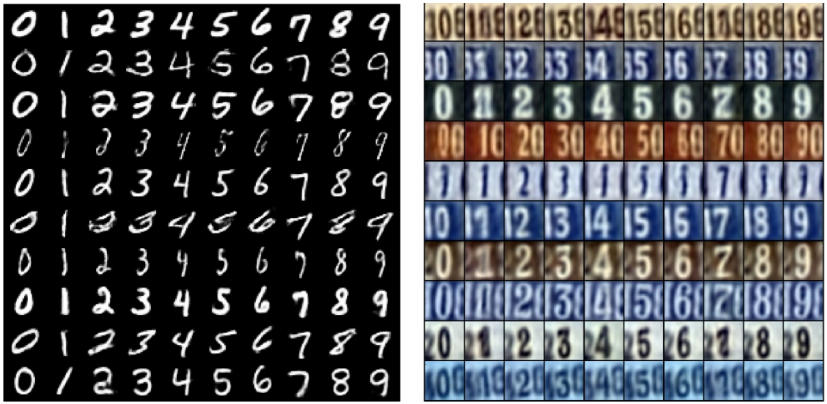
The figure is from Zhang et al. (2017); Prior: Results from a baseline

Class-Conditional Generation: Adversarial Autoencoder



The figure is from Makhzani et al. (2015).

Class-Conditional Generation: Adversarial Autoencoder



(a) MNIST (b) SVHN

Figure 7: Disentangling content and style (15-D Gaussian) on MNIST and SVHN datasets.

The figure is from Makhzani et al. (2015).

Class-Conditional Generation: Introducing Classifier

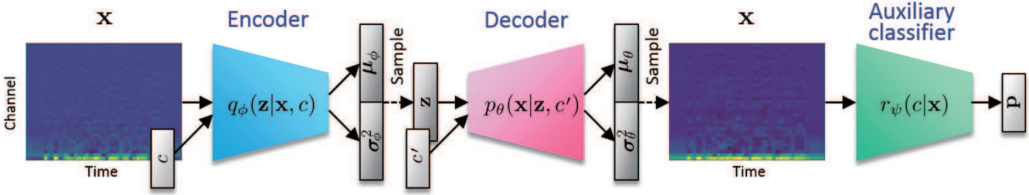


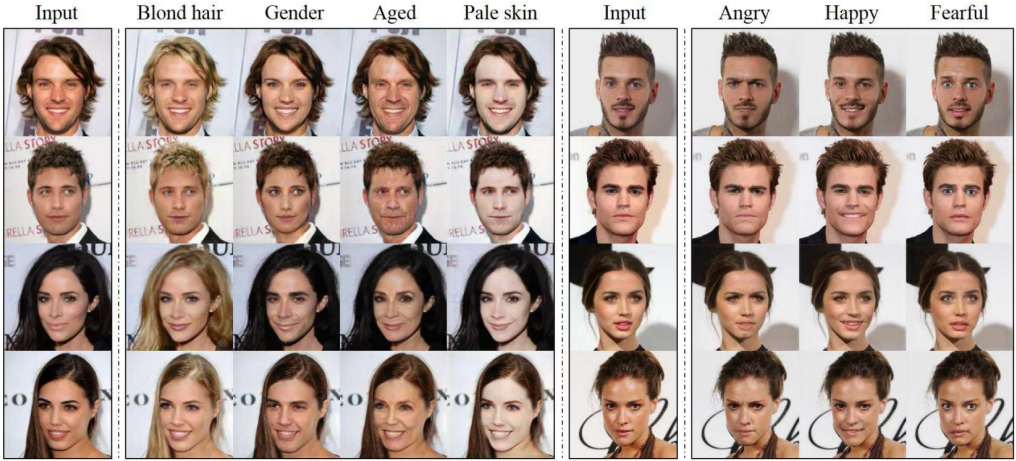
Figure: Architectures of encoder, generator (=decoder), and classifier networks in auxiliary classifier VAE (Kameoka et al., 2018). Both encoder and generator input domain labels. The auxiliary classifier is trained on real data and the generator is trained to minimize the sum of ELBO and classification loss of the generated data.

Data-to-Data Translation

Conditioning Data ($\vec{C}$)	Target Data ($\vec{X}$)	Name of corresponding task
Image	Image	Image-to-Image translation
Video	Video	Video-to-Video translation
Text	Text	Text-to-Text translation
Text	Image	Text-to-Image translation

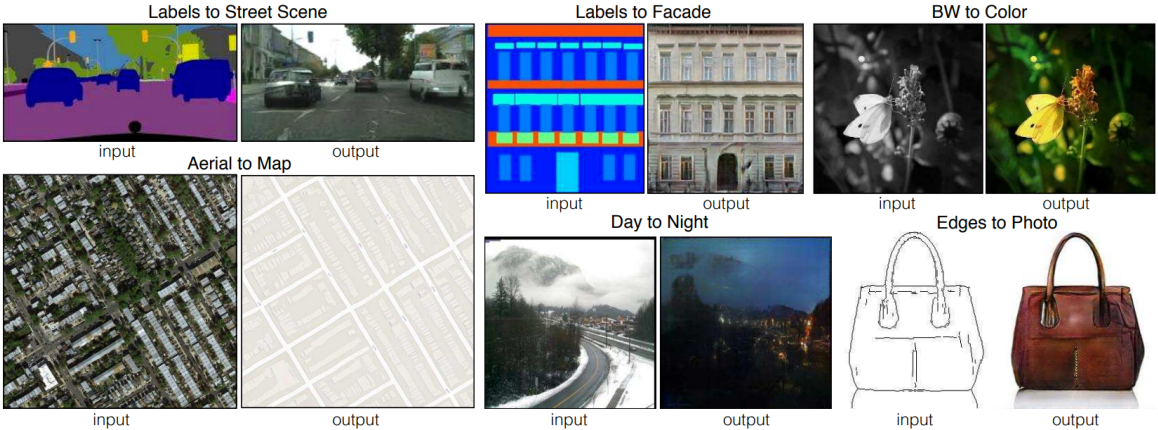
- Data-to-Data translation refers to synthesizing one type of data (from the *source domain*) using another (from the *target domain*).

Data-to-Data Translation: Image-to-Image Translation



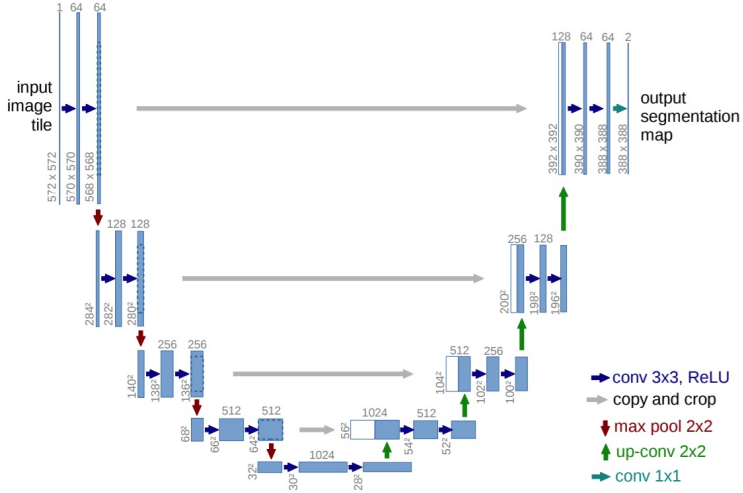
Images are from Choi et al. (2018).

Data-to-Data Translation: Image-to-Image Translation



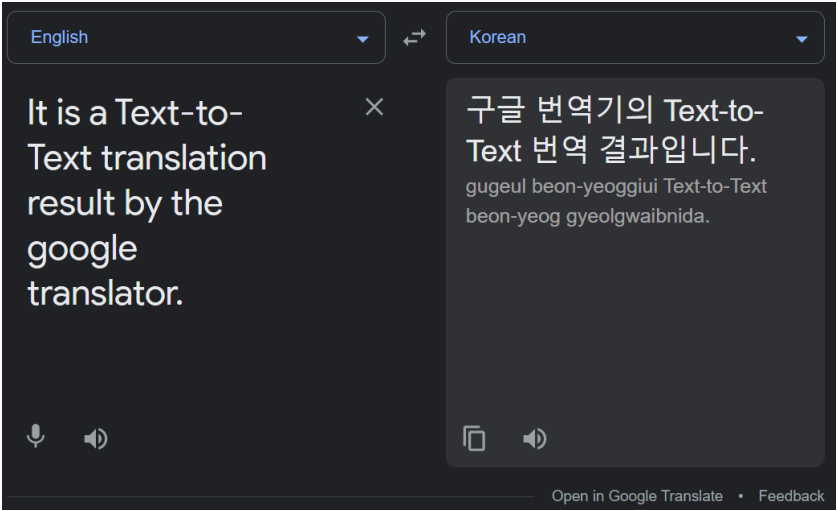
Summary of applications of Image-to-Image translation (Isola et al., 2017).

Data-to-Data Translation: Popular Network Architecture

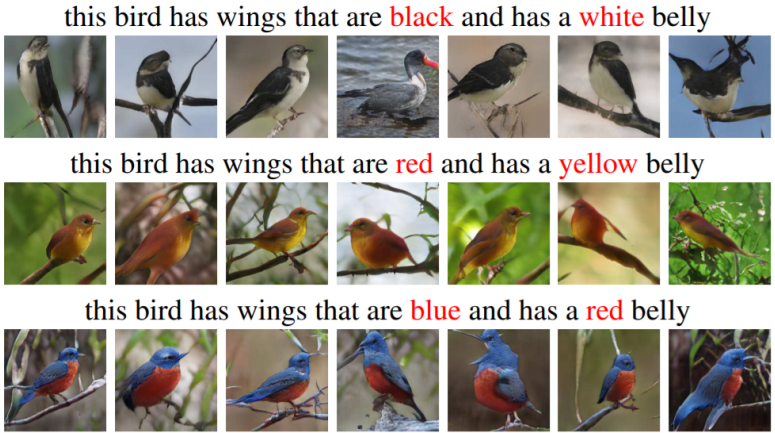


The figure is from Ronneberger et al. (2015).

Data-to-Data Translation: Text-to-Text Translation

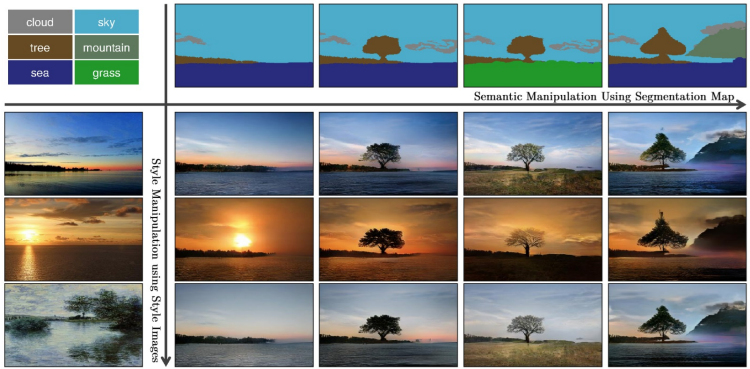


Data-to-Data Translation: Text-to-Image Translation



Images are from Xu et al. (2018).

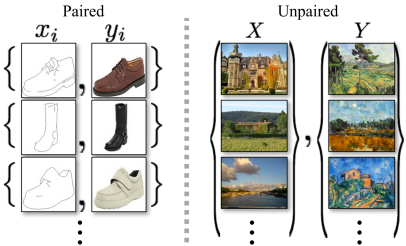
Data-to-Data Translation: Controllable Generation



- Data-to-Data Translation is an example of *Controllable Generation*. We can actively influence the detailed characteristics of the generated results by changing the conditioning data.

Images are from Park et al. (2019).

Data-to-Data Translation: Paired vs. Unpaired Datasets



- According to the presence of pair information, methods are divided into unpaired and paired settings.
- A dataset is *paired* if both source and target data (modalities) are observed for every subject, and pair information (ID of subjects) is provided. If the pair information is not provided, the dataset is *unpaired*.

The figure is from Zhu et al. (2017).

Data-to-Data Translation: What are Pairs?

- The pair of a given source datum usually indicates the most similar datum in the target dataset. The criterion of 'similar' depends on the context.
- Especially for unpaired datasets, it is difficult to define pairs. For example, when the source and target data are horse and zebra images, respectively, no object possesses the phenotype of both a horse and a zebra simultaneously.
- In this subsection, we review several of the earliest works, most of which assume a one-to-one relationship between source and target data, regardless of whether the training dataset contains the pair information.

Data-to-Data Translation: What are Pairs?

- In these earliest works, the translator is usually referred to as a "generator" and denoted by $G_\theta(\vec{C})$ since it learns the target data distribution ($p(\vec{x})$), but its role is more of a transformer than a generator.
- These are distinct from (class-)conditional models; they do not use real (high-dimensional) data such as images in generation and can generate new data corresponding to given conditioning data.
- However, the mapping from source data to target data is often one-to-many. For example, when we translate daytime scenes to their nighttime counterparts, there can be multiple nighttime scenes for a given daytime scene.
- Recent works have introduced latent factors $\vec{Z}$ to extend the translator $G_\theta(\vec{C})$ to $G_\theta(\vec{Z}, \vec{C})$. This research field is often called *multimodal synthesis*.

Paired Data-to-Data Translation: Pix2Pix

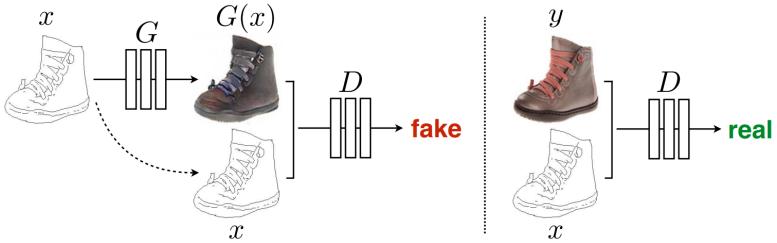


Figure: The architecture of Pix2Pix (Isola et al., 2017).

- We denote paired datasets by $\{(\vec{x}_i, \vec{c}_i)\}_{i=1}^n$. As in class-conditional generative models, the generator G_θ inputs $\vec{c}_i$ to synthesize datum similar to $\vec{x}_i$. This can be viewed as an multivariate regression utilizing adversarial learning.

Paired Data-to-Data Translation: Pix2Pix

- The goal is to learn a translator G_θ s.t. $G_\theta(\vec{C}) = \vec{X}$. For example, in Pix2Pix (Isola et al., 2017), the value function is:[2]

$$V(\theta, \phi) := \int \left(\log D_\phi(\vec{c}, \vec{x})\right) p_n(\vec{c}, \vec{x}) d\vec{c} d\vec{x} + \int \left(\log\left(1 - D_\phi(\vec{c}, G_\theta(\vec{c}))\right)\right) p_n(\vec{c}) d\vec{c}. \quad (26)$$

- The objective function is:

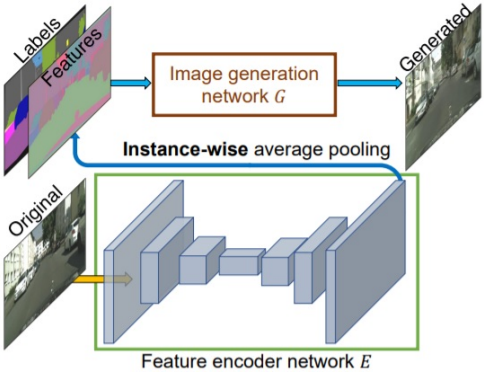
$$\int \|\vec{x} - G_\theta(\vec{c})\|_1 p_n(\vec{c}, \vec{x}) d\vec{c} d\vec{x} + \lambda V(\theta, \phi) \quad (27)$$

  where the first term is the prediction error and λ is a hyperparameter. It can be viewed as a regression error with adversarial learning-based regularization term.

- Note that the objective function includes $p_n(\vec{c}, \vec{x})$. We need paired source and target images to use it.

[2] The authors applied dropout, a stochastic layer, to build a probabilistic G_θ. For simplicity, we omit this stochastic part in the formulation in this lecture.

Paired Data-to-Data Translation: Pix2PixHD



- Pix2PixHD (Wang et al., 2018b) is an extension of Pix2Pix and introduces a feature encoder network to address multi-modal synthesis problem.

The figure is from Wang et al. (2018b).

Paired Data-to-Data Translation: Pix2PixHD

- The value function is:

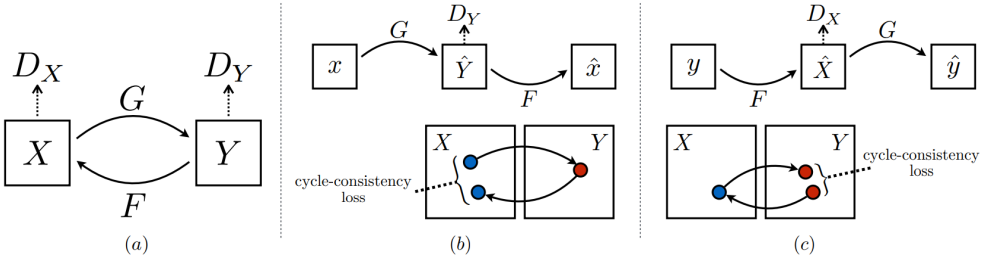
$$V(\theta, \phi, \psi) := \int \left(\log D_\phi(\vec{c}, \vec{x}) \right) p_n(\vec{c}, \vec{x}) d\vec{c} d\vec{x} + \int \left(\log \left(1 - D_\phi(\vec{c}, G_\theta(\vec{c}, E_\psi(\vec{x}))) \right) \right) p_n(\vec{c}, \vec{x}) d\vec{c} d\vec{x}. \tag{28}$$

- Pix2PixHD introduces latent variables, but their distribution is not estimated.
- The K-means clustering algorithm is applied to latent variables extracted from the training dataset. At inference time, one of K centroids is randomly sampled to generate target data.

Paired Data-to-Data Translation: Vid2Vid

Figure: Multi-modal synthesis results by Vid2Vid (Wang et al., 2018a).

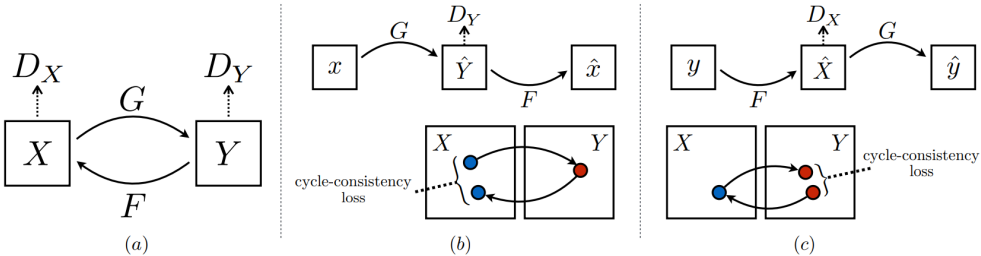
Unpaired Data-to-Data Translation: CycleGAN



- The pair information is usually not provided.
- CycleGAN (Zhu et al., 2017) is one of the earliest unpaired data-to-data translation models. It trains two translators and utilizes the cycle-consistency loss.

The figure is from Zhu et al. (2017).

Unpaired Data-to-Data Translation: CycleGAN



- CycleGANs have a pair of translators: one from the source domain to the target ($G_{C\to X,\theta}(\vec{c})$) and the other from the target back to the source ($G_{X\to C,\theta}(\vec{x})$).
- The goal is to find $(G_{C\to X,\theta}, G_{X\to C,\theta})$:
  1. (Transportation) $p_\theta(G_{C\to X,\theta}(\vec{c})) = p(\vec{x})$ and $p_\theta(G_{X\to C,\theta}(\vec{x})) = p(\vec{c})$.
  2. (Cycle-Consistency) $(G_{C\to X,\theta} \circ G_{X\to C,\theta})(\vec{c}) = \vec{c}$ and $(G_{X\to C,\theta} \circ G_{C\to X,\theta})(\vec{x}) = \vec{x}$.

The figure is from Zhu et al. (2017).

Unpaired Data-to-Data Translation: CycleGAN

- The value function can be expressed as:

$$V(\theta,\phi) := \left(\int \left(\log D_{X,\phi}(\vec{x}) \right) p_n(\vec{x}) d\vec{x} + \int \left(\log \left(1 - D_{X,\phi}(G_{C\to X,\theta}(\vec{c})) \right) \right) p_n(\vec{c}) d\vec{c} \right)$$
$$+ \left(\int \left(\log D_{C,\phi}(\vec{c}) \right) p_n(\vec{c}) d\vec{c} + \int \left(\log \left(1 - D_{C,\phi}(G_{X\to C,\theta}(\vec{x})) \right) \right) p_n(\vec{x}) d\vec{x} \right).$$

- They introduce cycle-consistency loss:

$$\int \|(G_{C\to X,\theta} \circ G_{X\to C,\theta})(\vec{c}) - \vec{c}\|_1 p_n(\vec{c}) d\vec{c} + \int \|(G_{X\to C,\theta} \circ G_{C\to X,\theta})(\vec{x}) - \vec{x}\|_1 p_n(\vec{x}) d\vec{x}.$$

  The cycle-consistency loss encourages the inverse relation between two translators.
- The loss function also includes the identity mapping loss term:

$$\int \|\vec{x} - G_{X\to C,\theta}(\vec{x})\| p_n(\vec{x}) d\vec{x} + \int \|\vec{c} - G_{C\to X,\theta}(\vec{c})\| p_n(\vec{c}) d\vec{c}.$$

  This approach can be viewed as learning pair information by neural networks.

Unpaired Data-to-Data Translation

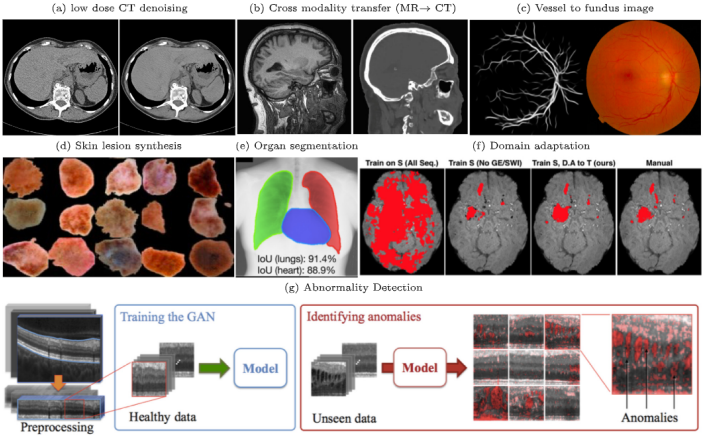


Figure: Important applications include image reconstruction, segmentation, detection, classification, and cross-modality synthesis (Yi et al., 2019)

Unpaired Data-to-Data Translation: Low Dose CT Denoising

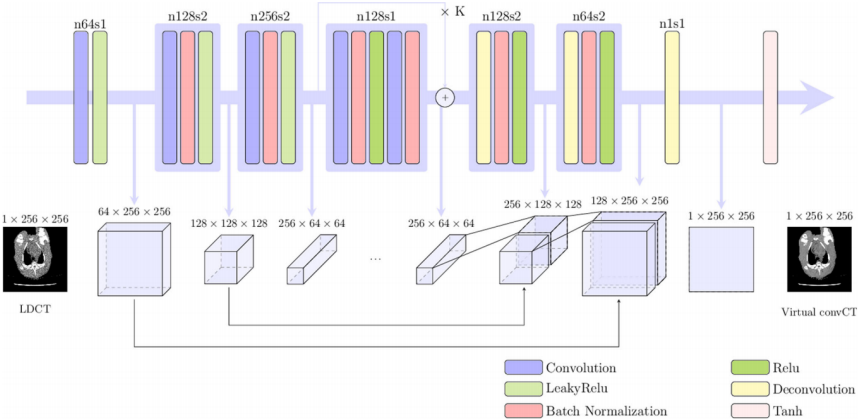


Figure: Low-dose CTs are feed-forwarded by a U-net to synthesize conventional CTs (Yi and Babyn, 2018). The dataset is a paired dataset.

Unpaired Data-to-Data Translation: Cross modality transfer (MR → CT)

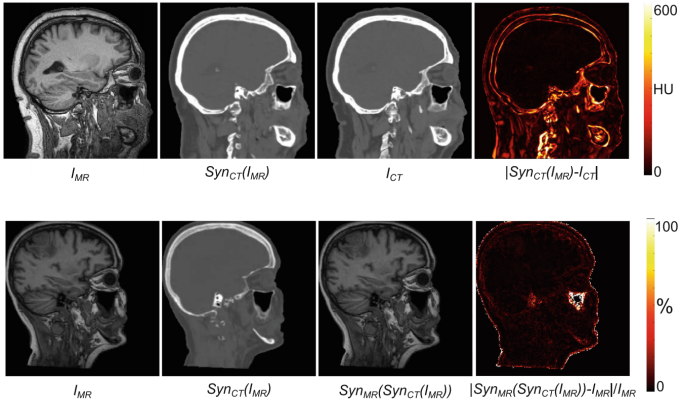


Figure: Wolterink et al. (2017) trained CycleGAN to translate MR images into CT images. The translation from CT to MR ($(2,2) \to (2,3)$) is also successful.

Unpaired Data-to-Data Translation: Cross modality transfer (Between MRIs)

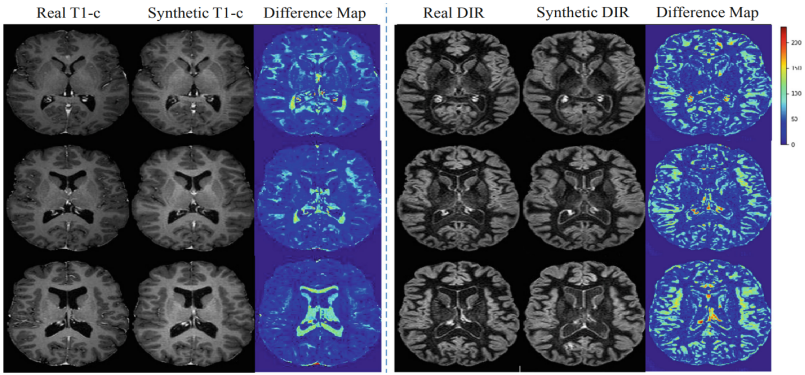


Figure: Li et al. (2019) trained a multi-CycleGAN between various modalities to translate MR images into other modalities. Left: $\{\text{T1}, \text{T2}, \text{Flair}\} \rightarrow$ Contrast-enhanced T1; Right: $\{\text{T1}, \text{T2}, \text{Flair}\} \rightarrow$ Double inversion recovery

Unpaired Data-to-Data Translation: Organ Segmentation

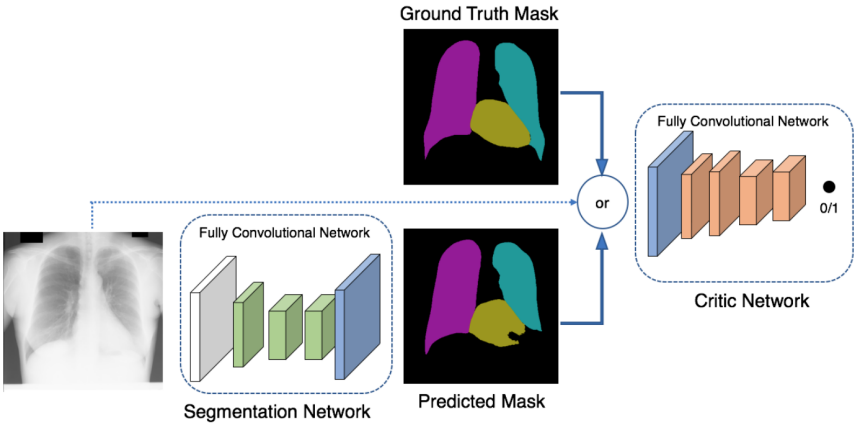


Figure: Dai et al. (2018) viewed the segmentation model as a conditional image synthesis model and used the summation of segmentation loss and adversarial loss as objective. The proposed model has better segmentation performance than models without the adversarial loss.

Unpaired Data-to-Data Translation: Anomaly Detection

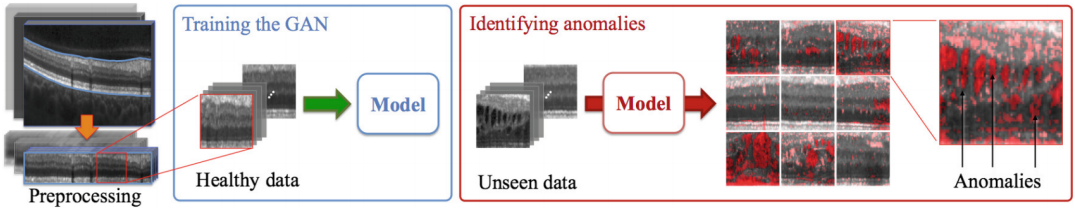


Figure: Schlegl et al. (2017) first trained unconditional image synthesis model with only healthy data, and then use it to calculate anomaly scores. Their anomaly scores are the summation of reconstruction error in raw data space and extracted feature space, respectively.

Outline

Mode Collapse

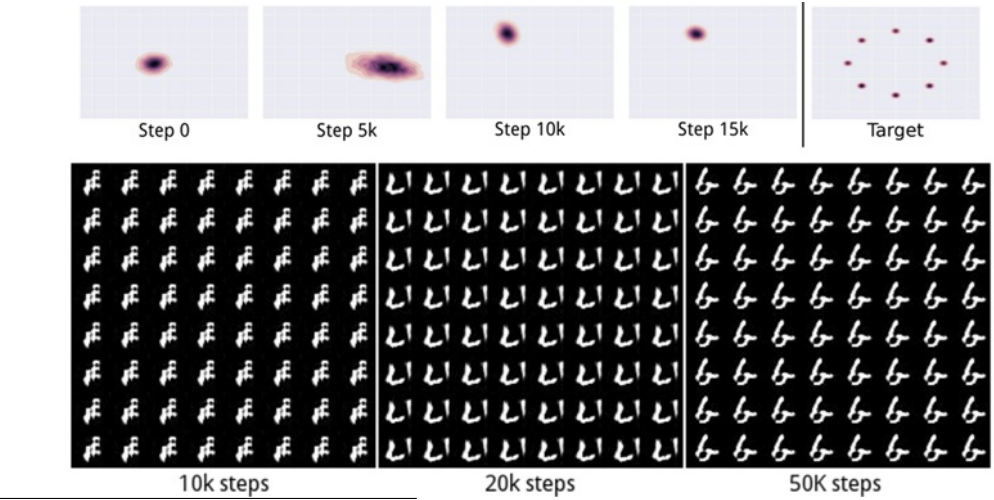
- Again, the adversarial learning process involves alternately maximizing and minimizing the negative cross-entropy loss:

$$\theta^* \in \arg\min_{\theta} \Big(\max_{\phi} V(\theta, \phi) \Big). \tag{29}$$

  This process seeks to find the best generator parameters θ while considering the optimal responses of the discriminator.

- For a given generator parameter θ, GANs approximate $\max_{\phi} V(\theta, \phi)$ by updating the discriminator parameter ϕ. However, there is usually a substantial gap between the theoretical maximum $\max_{\phi} V(\theta^{(t)}, \phi)$ and the value obtained at any particular iteration $V(\theta^{(t)}, \phi^{(t)})$.

Mode Collapse



Images are from Metz et al. (2017).

Mode Collapse: Unrolled GAN

- Unrolled GANs (Metz et al., 2017) aggressively update the discriminator parameters:

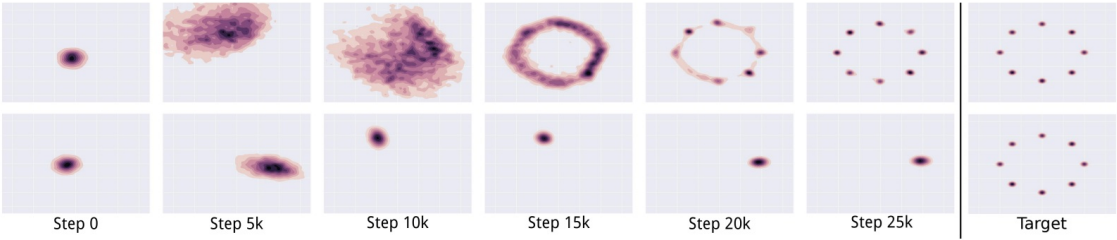
$$\phi^{(t,k+1)} = \phi^{(t,k)} + \eta^{(k)} \frac{dV(\theta^{(t)}, \phi)}{d\phi}\Big|_{\phi=\phi^{(t,k)}} \tag{30}$$

  where $\eta^{(k)}$ represents the learning rate. By repeating this process K times, we may expect that $\phi^{(t,K)}$ better approximates the local maximum of $V(\theta^{(t)}, \phi)$.

- The generator parameter is then updated via:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{dV(\theta, \phi^{(t,K)})}{d\theta}\Big|_{\theta=\theta^{(t)}} \tag{31}$$

Mode Collapse: Unrolled GAN



Images are from Metz et al. (2017).

Mode Collapse: Other Remedies

- Other remedies include modifying the value function.
- Popular approaches include Wasserstein generative models. These will be reviewed in detail in a later section on optimal transport-based approaches.

Outline

f-GAN

- We have reviewed the following relationship in GANs, which holds up to a constant addition and sign-preserving multiplication: Using discriminator networks parameterized with ϕ,

$$\mathrm{JS}(p_n \parallel p_\theta) \approx V(\theta, \hat{\phi}_n(\theta)).$$

- Nowozin et al. (2016) generalized the concept of using auxiliary networks to approximate other f-divergences:

$$\mathcal{D}_f(p||q) := \int f\left(\frac{p(\vec{x})}{q(\vec{x})}\right) q(\vec{x}) d\vec{x} \tag{32}$$

  where $f : \mathbb{R}^+ \to \mathbb{R}$ is a convex function satisfying $f(1) = 0$.

- The key idea is to introduce the convex conjugate function (or Fenchel conjugate, Hiriart-Urruty and Lemaréchal, 2004) to derive variational estimations of f-divergences.

f-Divergence

- Examples include KL divergence, total variation distance, and Jensen-Shannon (JS) divergence.
- **KL divergence**: $\mathrm{KL}(p||q) := \int \log(p(\vec{x})/q(\vec{x}))p(\vec{x})d\vec{x}$
  Proof: When $f(u) = u\log u$, $\mathcal{D}_f(p||q) = \int \left(\frac{p(\vec{x})}{q(\vec{x})}\log\frac{p(\vec{x})}{q(\vec{x})}\right)q(\vec{x})d\vec{x} = \int \left(\log\frac{p(\vec{x})}{q(\vec{x})}\right)p(\vec{x})d\vec{x}$.
- **Total variation distance**: $\delta(p,q) := \frac{1}{2}\int\left|p(\vec{x}) - q(\vec{x})\right| d\vec{x}$
  Proof: When $f(u) = |u-1|/2$, $\mathcal{D}_f(p||q) = \frac{1}{2}\int\left|\frac{p(\vec{x})}{q(\vec{x})} - 1\right| q(\vec{x})d\vec{x} = \frac{1}{2}\int\left|p(\vec{x}) - q(\vec{x})\right| d\vec{x}$.

f-Divergence

- **Jensen-Shannon divergence**:

$$\mathrm{JS}(p||q) := \frac{1}{2}\Big(\mathrm{KL}(p||\frac{p+q}{2}) + \mathrm{KL}(q||\frac{p+q}{2})\Big) \tag{33}$$

Proof: When $f(u) = \frac{1}{2}\Big(- (u+1)\log\frac{1+u}{2} + u\log u\Big)$, $\mathcal{D}_f(p||q)$ can be expressed as

$$\begin{aligned}
&\frac{1}{2}\int\Big(- \Big(\frac{p(\vec{x})}{q(\vec{x})}+1\Big)\log\frac{1+p(\vec{x})/q(\vec{x})}{2} + \frac{p(\vec{x})}{q(\vec{x})}\log\frac{p(\vec{x})}{q(\vec{x})}\Big)q(\vec{x})d\vec{x}\\
&= \frac{1}{2}\int\Big(- \Big(p(\vec{x})+q(\vec{x})\Big)\Big(\log\frac{p(\vec{x})+q(\vec{x})}{2} - \log q(\vec{x})\Big) + p(\vec{x})\Big(\log p(\vec{x}) - \log q(\vec{x})\Big)\Big)d\vec{x}\\
&= \frac{1}{2}\int\Big(\Big(\log p(\vec{x}) - \log\frac{p(\vec{x})+q(\vec{x})}{2}\Big)p(\vec{x}) + \Big(\log q(\vec{x}) - \log\frac{p(\vec{x})+q(\vec{x})}{2}\Big)q(\vec{x})\Big)d\vec{x}.
\end{aligned} \tag{34}$$

f-Divergence

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

List of popular examples of f-divergences, edited from Nowozin et al. (2016).

f-GAN

- We denote the convex conjugates of functions f by $f^*(t) := \sup_u \{ut - f(u)\}$.
- The f^* relates f and its subgradients. When f is convex and differentiable[3], the following properties hold:
  1. f^* is also convex and differentiable.
  2. Duality holds, i.e., $(f^*)^* = f$.
  3. The relation $f(u) + f^*(t) = ut$ holds if and only if $t = f'(u)$.
- When f' is invertible, $f^*(t) = (f')^{-1}(t)t - (f \circ f'^{-1})(t)$.

[3] For more general functions, check Hiriart-Urruty and Lemaréchal (2004).

f-GAN

- By duality and the definition of supremum, we have the following variational formulation:

$$\begin{aligned}\mathcal{D}_f(p_n||p_\theta) &= \int \sup_t \left\{ \frac{p_n(\vec{x})}{p_\theta(\vec{x})} t - f^*(t) \right\} p_\theta(\vec{x}) d\vec{x} \\ &\geq \sup_{T_\phi} \left(\int T_\phi(\vec{x}) p_n(\vec{x}) d\vec{x} - \int f^*(T_\phi(\vec{x})) p_\theta(\vec{x}) d\vec{x} \right).\end{aligned} \tag{35}$$

- We assume that f is differentiable and that $\{T_\phi | \phi \in \Phi\}$ is sufficiently flexible such that for any given θ, there exists $\phi^*(\theta)$ where:

$$T_{\phi^*(\theta)}(\vec{x}) = f'(p_n(\vec{x})/p_\theta(\vec{x})). \tag{36}$$

This satisfies the equality condition of Equation (35).

f-GAN

- Define $F(\theta, \phi) := \int T_\phi(\vec{x}) p_n(\vec{x})\, d\vec{x} - \int f^*\Big(T_\phi(\vec{x})\Big) p_\theta(\vec{x})\, d\vec{x}$ and $\theta^* := \arg\min_\theta \Big(\max_\phi F(\theta, \phi)\Big)$.
- Then, $\max_\phi F(\theta, \phi) = F(\theta, \phi^*(\theta)) = \mathcal{D}_f(p_n \| p_\theta)$. Thus, p_{θ^*} is a minimizer of the f-divergence.
- **Example 1 (KL divergence)**: Let $f(u) = u \log u$ and $f^*(t) = \exp(t-1)$. We can express $F(\theta, \phi)$ as follows:

$$F(\theta, \phi) = \int T_\phi(\vec{x}) p_n(\vec{x})\, d\vec{x} - \int \exp\Big(T_\phi(\vec{x}) - 1\Big) p_\theta(\vec{x})\, d\vec{x}. \tag{37}$$

f-GAN

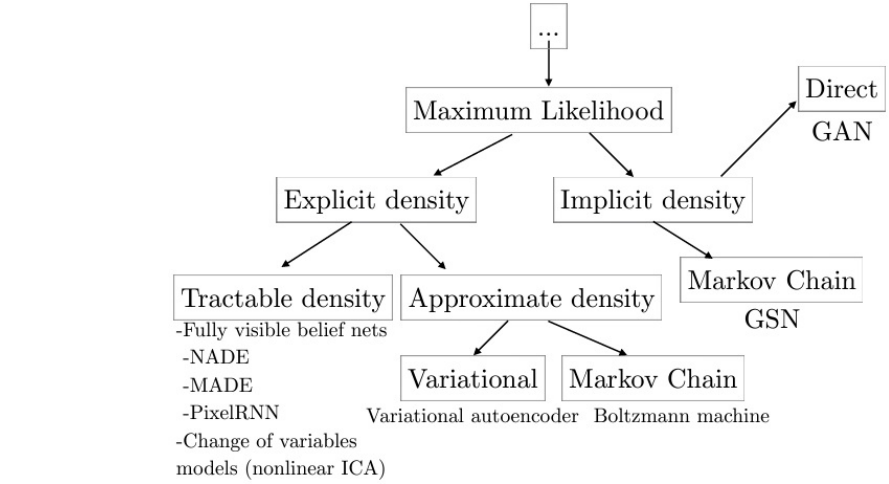
- **Example 2 (JS divergence)**: Let $f(u) = -(u+1)\log\frac{1+u}{2} + u\log u$ and $f^*(t) = -\log(2-\exp(t))$. We can express $F(\theta,\phi)$ as follows:

$$F(\theta,\phi) = \int T_\phi(\vec{x}) p_n(\vec{x})\, d\vec{x} + \int \log\left(2 - \exp\left(T_\phi(\vec{x})\right)\right) p_\theta(\vec{x})\, d\vec{x}. \tag{38}$$

- GANs are special cases of f-GANs. When we model $T_\phi(\vec{x}) = \log D_\phi(\vec{x}) + \log 2$, $F(\theta,\phi) = V(\theta,\phi) + \log 4$, and $T_{\phi^*(\theta)}(\vec{x}) = \log D_{\phi^*(\theta)}(\vec{x}) + \log 2 = \log\frac{p_n(\vec{x})}{p_n(\vec{x})+p_\theta(\vec{x})} + \log 2$ hold.

Outline

Revisit: A Taxonomy of Deep Generative Models



The figure is from Goodfellow (2016).

References I

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. (2018). Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dai, W., Dong, N., Wang, Z., Liang, X., Zhang, H., and Xing, E. P. (2018). Scan: Structure correcting adversarial network for organ segmentation in chest x-rays. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 263–273. Springer.

Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

References II

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2016). Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2004). *Fundamentals of convex analysis*. Springer Science & Business Media.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

References III

Kameoka, H., Kaneko, T., Tanaka, K., and Hojo, N. (2018). Acvae-vc: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder. *arXiv preprint arXiv:1808.05092*.

Li, C., Liu, H., Chen, C., Pu, Y., Chen, L., Henao, R., and Carin, L. (2017). Alice: Towards understanding adversarial learning for joint distribution matching. *Advances in neural information processing systems*, 30.

Li, H., Paetzold, J. C., Sekuboyina, A., Kofler, F., Zhang, J., Kirschke, J. S., Wiestler, B., and Menze, B. (2019). Diamondgan: unified multi-modal generative adversarial networks for mri sequences synthesis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 795–803. Springer.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

References IV

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. In *International Conference on Learning Representations*.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Niu, Z., Zhou, M., Wang, L., Gao, X., and Hua, G. (2016). Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4920–4928.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR.

References V

Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019). Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *International conference on learning representations*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer.

References VI

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schölkopf, B. (2018). Wasserstein auto-encoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Liu, G., Tao, A., Kautz, J., and Catanzaro, B. (2018a). Video-to-video synthesis. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1152–1164.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018b). High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807.

References VII

Wolterink, J. M., Dinkla, A. M., Savenije, M. H., Seevinck, P. R., van den Berg, C. A., and Išgum, I. (2017). Deep mr to ct synthesis using unpaired data. In *International workshop on simulation and synthesis in medical imaging*, pages 14–23. Springer.

Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., and He, X. (2018). Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324.

Yi, X. and Babyn, P. (2018). Sharpness-aware low-dose ct denoising using conditional generative adversarial network. *Journal of digital imaging*, 31(5):655–669.

Yi, X., Walia, E., and Babyn, P. (2019). Generative adversarial network in medical imaging: A review. *Medical image analysis*, 58:101552.

References VIII

Zenati, H., Romain, M., Foo, C.-S., Lecouat, B., and Chandrasekhar, V. (2018). Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE.

Zhang, Z., Song, Y., and Qi, H. (2017). Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.