Probabilistic Graphical Models

Inference & Learning in DL

Zhiting Hu

Lecture 19, March 29, 2017

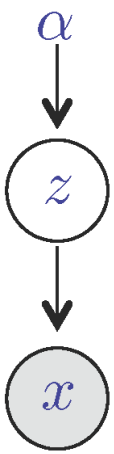
Reading:

Deep Generative Models

- Explicit probabilistic models
  - Provide an explicit parametric specification of the distribution of $\boldsymbol{x}$
  - Tractable likelihood function $\mathrm{p}_{\theta}(\boldsymbol{x})$

  - E.g.,

$$p(x, z|\alpha) = p(x|z)p(z|\alpha)$$



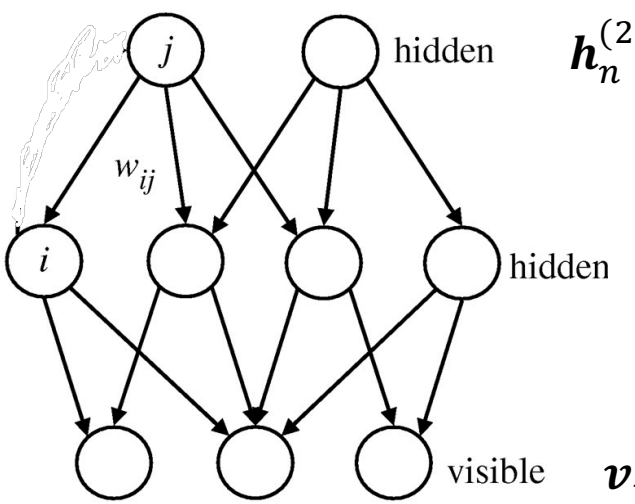
Deep Generative Models

- Explicit probabilistic models
  - Provide an explicit parametric specification of the distribution of $\boldsymbol{x}$
  - Tractable likelihood function $\mathrm{p}_\theta(\boldsymbol{x})$

  - E.g., Sigmoid Belief Nets

$$p\left(v_{kn}=1 \middle| \boldsymbol{w}_k, \boldsymbol{h}_n^{(1)}, c_k\right) = \sigma\left(\boldsymbol{w}_k^T \boldsymbol{h}_n^{(1)} + c_k\right)$$

$$p\left(h_{in}^{(1)}=1 \middle| \boldsymbol{w}_i, \boldsymbol{h}_n^{(2)}, c_i\right) = \sigma(\boldsymbol{w}_i^T \boldsymbol{h}_n^{(2)} + c_i)$$



$$\boldsymbol{h}_n^{(2)} = \{0,1\}^J$$

$$\boldsymbol{h}_n^{(1)} = \{0,1\}^I$$

$$\boldsymbol{v}_n = \{0,1\}^K$$

Deep Generative Models

- Explicit probabilistic models
  - Provide an explicit parametric specification of the distribution of $\boldsymbol{x}$
  - Tractable likelihood function $\mathrm{p}_\theta(\boldsymbol{x})$
  - E.g., Deep generative model parameterized with NNs (e.g., VAEs)

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}; \mu_\theta(\boldsymbol{z}), \sigma)$$

$$p(\boldsymbol{z}) = N(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$$

Deep Generative Models

- Implicit probabilistic models
  - Defines a stochastic process to simulate data $\boldsymbol{x}$
  - Do not require tractable likelihood function
  - Data simulator
  - Natural approach for problems in population genetics, weather, ecology, etc.

  - E.g., generate data from a deterministic equation given parameters and random noise (e.g., GANs)

$$\boldsymbol{x}_n = g(\boldsymbol{z}_n; \boldsymbol{\theta})$$

$$\boldsymbol{z}_n \sim N(\boldsymbol{0}, \boldsymbol{I})$$

Recap: Variational Inference

- Consider a probabilistic model $p_\theta(\boldsymbol{x}, \boldsymbol{z})$
- Assume variational distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$
- Lower bound for log likelihood

$$
\begin{aligned}
&\log p(\boldsymbol{x}) \\
&= KL\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\Big) + \int_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \\
&\geq \int_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \\
&:= \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})
\end{aligned}
$$

- Free energy

$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

Wake Sleep Algorithm

- Consider a generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$
  - E.g., sigmoid brief nets
- Variational bound:

$$\log p(\boldsymbol{x}) \geq \int_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} := \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})$$

- Use a inference network $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
- Maximize the bound w.r.t. p_θ → *Wake* phase
  - $\max_{\boldsymbol{\theta}} \mathrm{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$
  - Get samples from $q(\boldsymbol{z}|\boldsymbol{x})$ through bottom-up pass
  - Use the samples as targets for updating the generator

Wake Sleep Algorithm

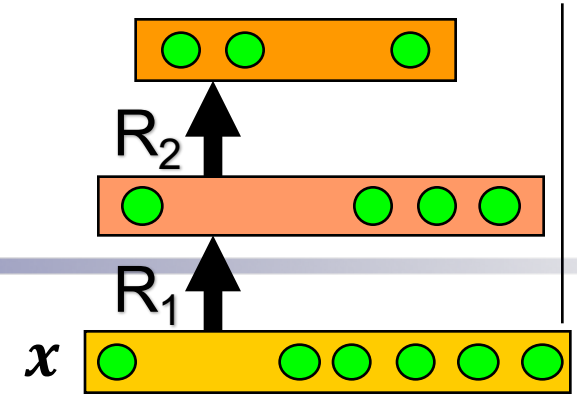
- [Hinton et al., Science 1995]
- Generally applicable to a wide range of generative models by training a separate inference network

- Consider a generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, with prior $p(\boldsymbol{z})$
  - E.g., multi-layer brief nets
- Free energy

$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- Inference network $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - a.k.a. recognition network

Wake Sleep Algorithm



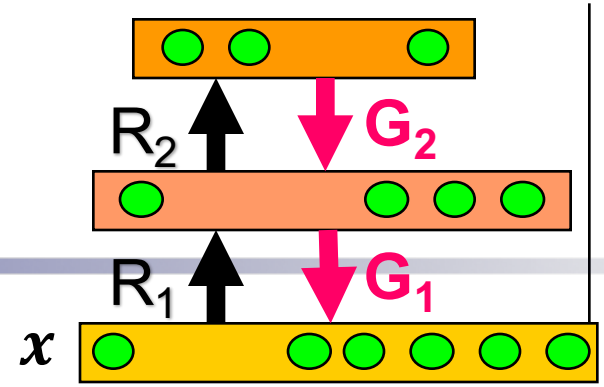
- Free energy:

$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- Minimize the free energy w.r.t. p_θ → *Wake* phase
  - $\max_{\boldsymbol{\theta}} \mathrm{E}_{q(\boldsymbol{z}|\boldsymbol{x})} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$
  - Get samples from q through bottom-up pass on training data
  - Use the samples as targets for updating the generator

[Figure courtesy: Maei's slides]

Wake Sleep Algorithm



- Free energy:

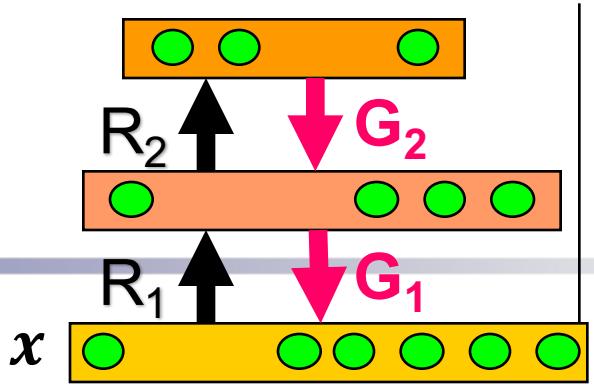
$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- Maximize the free energy w.r.t. $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$?
  - computationally expensive / high variance
- Instead, maximize w.r.t. $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ → *Sleep* phase

$$F'(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(p(\boldsymbol{z}|\boldsymbol{x}) \,||\, q_{\phi}(\boldsymbol{z}|\boldsymbol{x}))$$

  - $\max_{\boldsymbol{\phi}} \mathrm{E}_{p(\boldsymbol{z},\boldsymbol{x})} \left[\log q_{\phi}(\boldsymbol{z}|\boldsymbol{x})\right]$
  - "Dreaming" up samples from p through top-down pass
  - Use the samples as targets for updating the recognition network

Wake Sleep Algorithm



- Wake phase:
  - Use recognition network to perform a bottom-up pass in order to create samples for layers above (from data)
  - Train generative network using samples obtained from recognition model
- Sleep phase:
  - Use generative weights to reconstruct data by performing a top-down pass
  - Train recognition weights using samples obtained from generative model

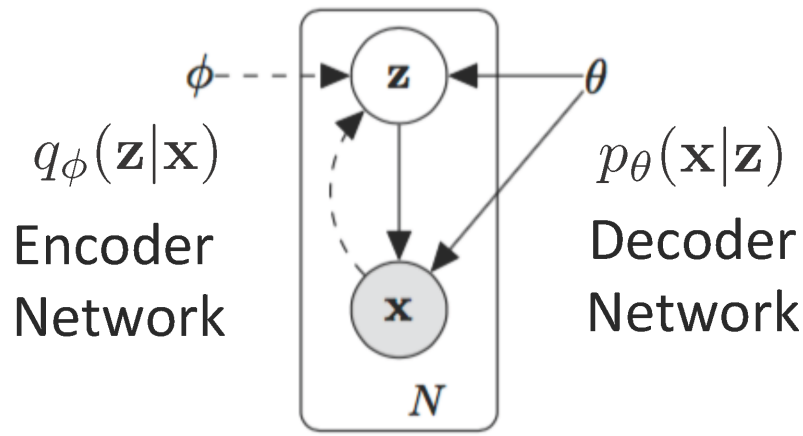
- KL is not symmetric
- Doesn't optimize a well-defined objective function
- Not guaranteed to converge

Variational Auto-encoders (VAEs)

- [Kingma & Welling, 2014]
- Enjoy similar applicability with wake-sleep algorithm
  - Not applicable to discrete latent variables
- Optimize a variational lower bound on the log-likelihood
- Reduce variance through reparameterization of the recognition distribution
  - Alternatives: use control variates as in reinforcement learning [Mnih & Gregor, 2014]

Variational Auto-encoders (VAEs)

- Generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, with prior $p(\boldsymbol{z})$
  - a.k.a. decoder
- Inference network $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - a.k.a. encoder, recognition network



- Variational lower bound

$$\log p(\boldsymbol{x}) \geq \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x},\boldsymbol{z})] - \mathrm{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})\right)$$
$$:= \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})$$

Variational Auto-encoders (VAEs)

- Variational lower bound

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$$

- Optimize $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ w.r.t. $p_\theta(\boldsymbol{x}|\boldsymbol{z})$
  - the same with the wake phase
- Optimize $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ w.r.t. $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - Directly computing the gradient with MC estimation

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z})] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z})\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$$

a REINFORCE-like update rule which suffers from high variance [Mnih & Gregor 2014]

(Next lecture for more on REINFORCE)

  - VAEs use a reparameterization trick to reduce variance

VAEs: Reparameterization Trick

- $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ $\Rightarrow$

  auxiliary variable

  $\epsilon \sim p(\epsilon)$

  deterministic variable

  $\mathbf{z} = g_\phi(\epsilon, \mathbf{x})$

Example:

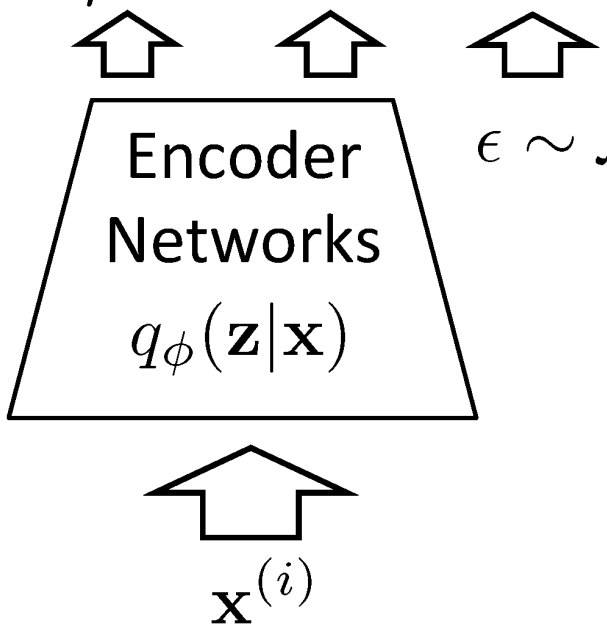
$z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$ $\Rightarrow$ $\epsilon \sim \mathcal{N}(0, 1)$

$z = \mu + \sigma\epsilon$

VAEs: Reparameterization Trick

- $$q_\phi\left(\boldsymbol{z}^{(i)}\middle|\boldsymbol{x}^{(i)}\right) = \mathcal{N}(\boldsymbol{z}^{(i)}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I})$$

$$\mathbf{z}^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}$$



$$\epsilon \sim \mathcal{N}(0, \mathbf{I})$$

$\boldsymbol{z} = \boldsymbol{z}_\phi(\boldsymbol{\epsilon})$ is a deterministic mapping of $\boldsymbol{\epsilon}$

[Figure courtesy: Chang's slides]

VAEs: Reparameterization Trick

- Variational lower bound

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathrm{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})\right)$$

$$\mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] = \mathrm{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\log p_\theta\left(\boldsymbol{x}, \boldsymbol{z}_\phi(\boldsymbol{\epsilon})\right)\right]$$

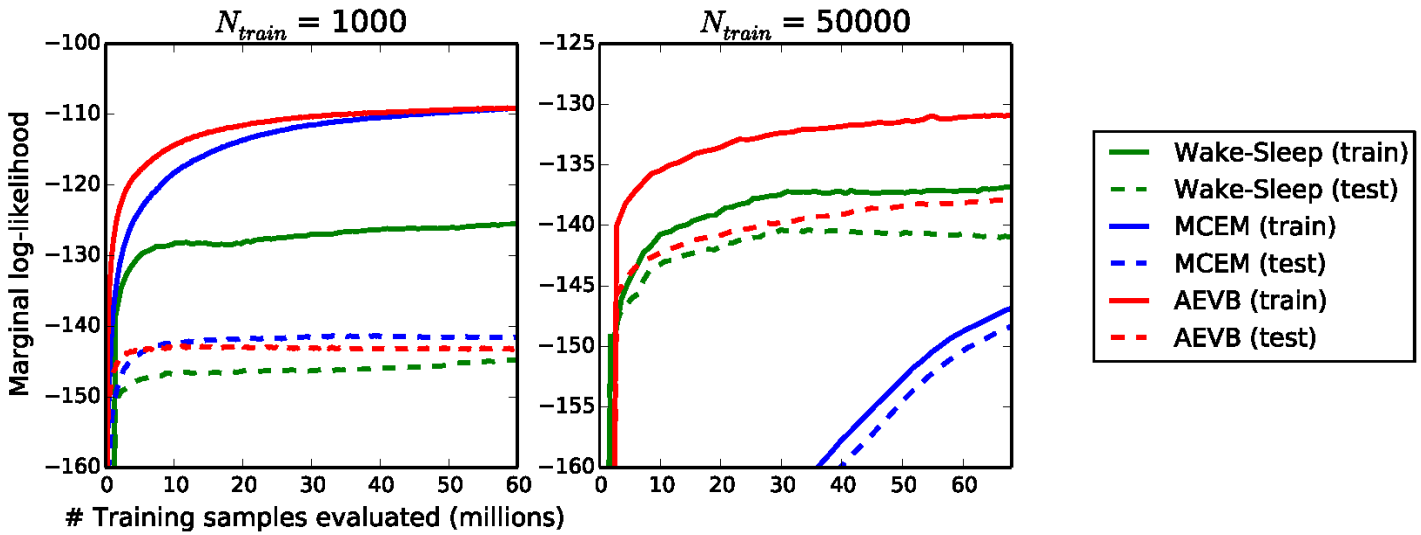
- Optimize $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ w.r.t. $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - $\nabla_\phi \, \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] = \mathrm{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\nabla_\phi \log p_\theta\left(\boldsymbol{x}, \boldsymbol{z}_\phi(\boldsymbol{\epsilon})\right)\right]$
  - Uses the gradients w.r.t. the latent variables
  - For Gaussian distributions, $\mathrm{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})\right)$ can be computed and differentiated analytically

VAEs: Training

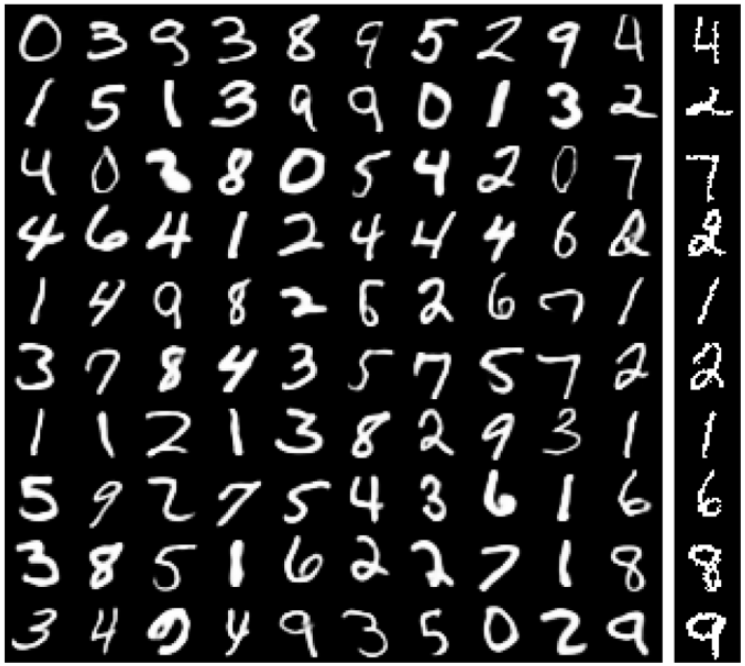
Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

VAEs: Results

- 

VAEs: Results



Generated MNIST images

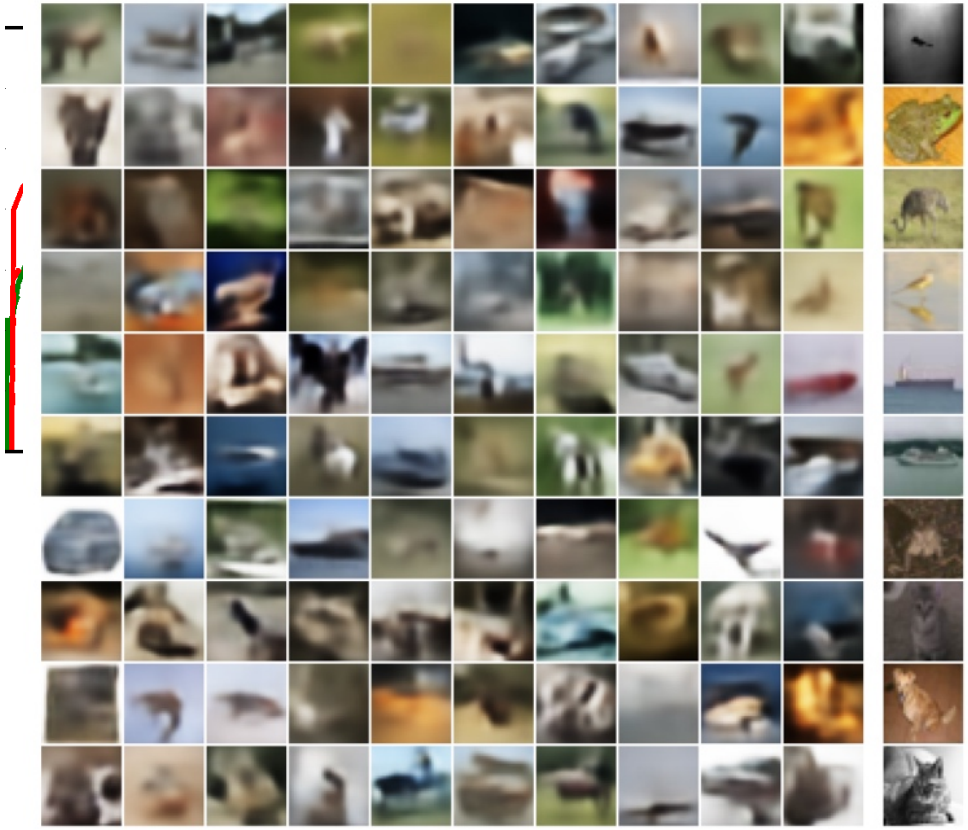
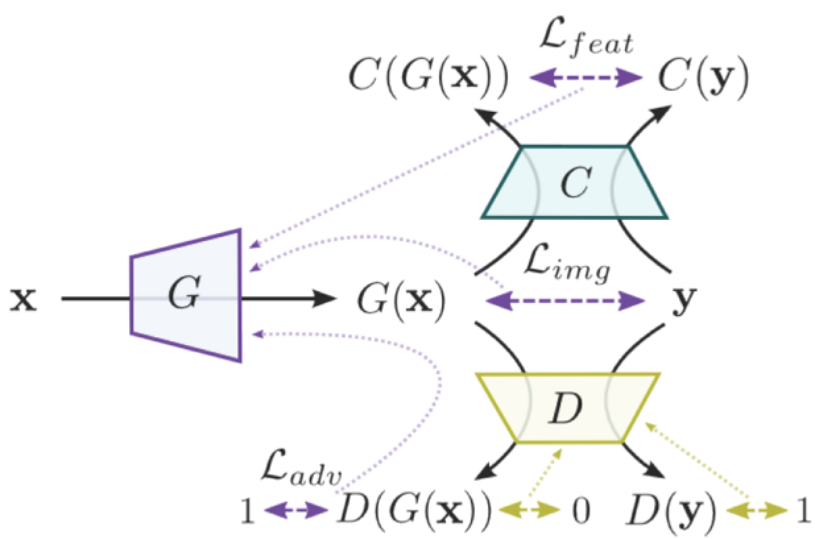


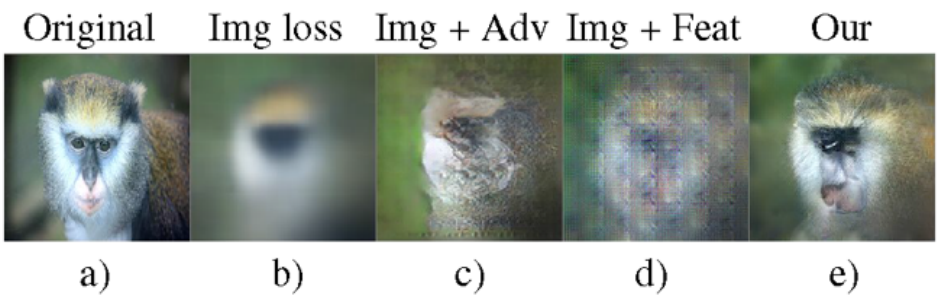
Figure 12. **Generated CIFAR images.** The rightmost column shows the nearest training examples to the column beside it.

[Gregor et al., 2015]

VAEs: Limitations and variants

- Element-wise reconstruction error
  - For image generation, to reconstruct every pixels
  - Sensitive to irrelevant variance, e.g., translations
  - Variant: feature-wise (perceptual-level) reconstruction [Dosovitskiy et al., 2016]
    - Use a pre-trained neural network to extract features of data
    - Generated images are required to have similar feature vectors with the data
  - Variant: Combining VAEs with GANs [Larsen et al., 2016] (more later)





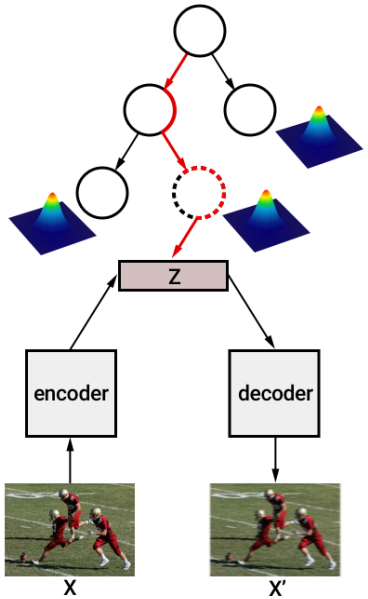
Reconstruction results with different loss

VAEs: Limitations and variants

- Not applicable to discrete latent variables
  - Differentiable reparameterization does not apply to discrete variables
  - Wake-sleep algorithm/GANs allow discrete latents
  - Variant: marginalize out discrete latents [Kingma et al., 2014]
    - Expensive when the discrete space is large
  - Variant: use continuous approximations
    - Gumbel-softmax [Jang et al, 2017] for approximating multinomial variables
  - Variant: combine VAEs with wake-sleep algorithm [Hu et al., 2017]

VAEs: Limitations and variants

- Usually use a fixed standard normal distribution as prior
  - $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z};\ \boldsymbol{0}, \boldsymbol{I})$
  - For ease of inference and learning
  - Limited flexibility: converting the data distribution to fixed, single-mode prior distribution
  - Variant: use hierarchical nonparametric priors [Goyal et al., 2017]
    - E.g., Dirichlet process, nested Chinese restaurant process (more later)
    - Learn the structures of priors jointly with the model



VAEs: Limitations and variants

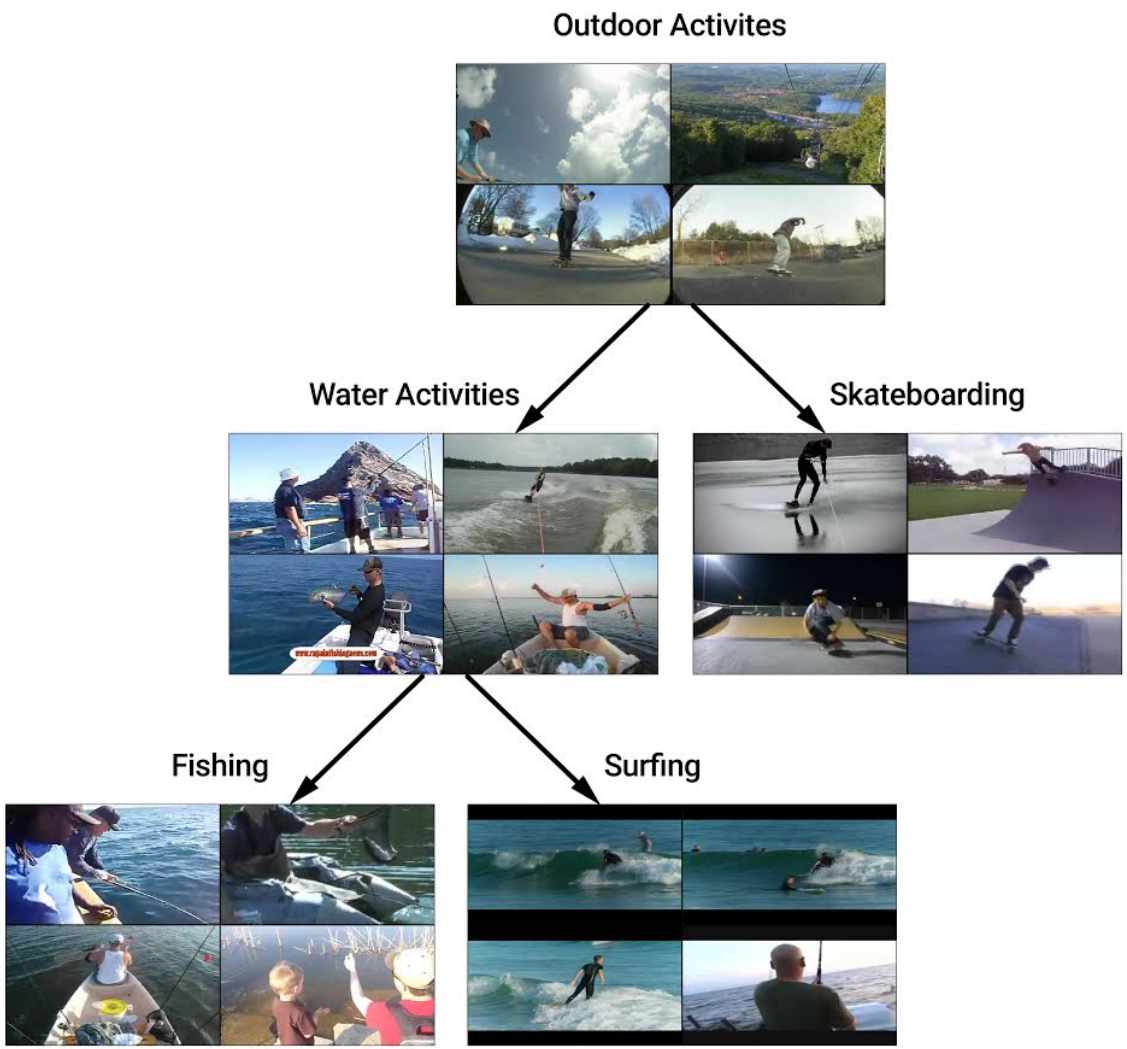
ion as prior

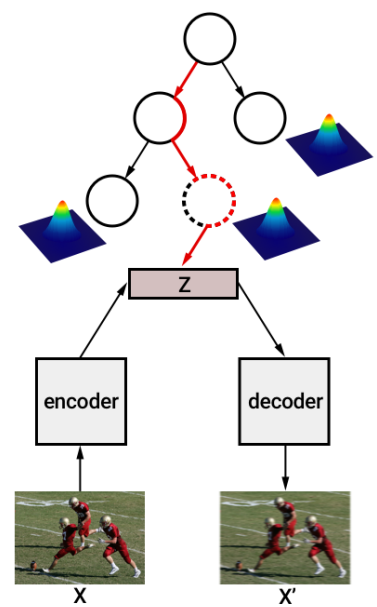
fixed, single-mode prior

al et al., 2017]

nt process (more later)

del





Deep Generative Models

- Implicit probabilistic models
  - Defines a stochastic process to simulate data $\boldsymbol{x}$
  - Do not require tractable likelihood function
  - Data simulator
  - Natural approach for problems in population genetics, weather, ecology, etc.

  - E.g., generate data from a deterministic equation given parameters and random noise (e.g., GANs)

$$\boldsymbol{x}_n = g(\boldsymbol{z}_n; \boldsymbol{\theta})$$

$$\boldsymbol{z}_n \sim N(\boldsymbol{0}, \boldsymbol{I})$$

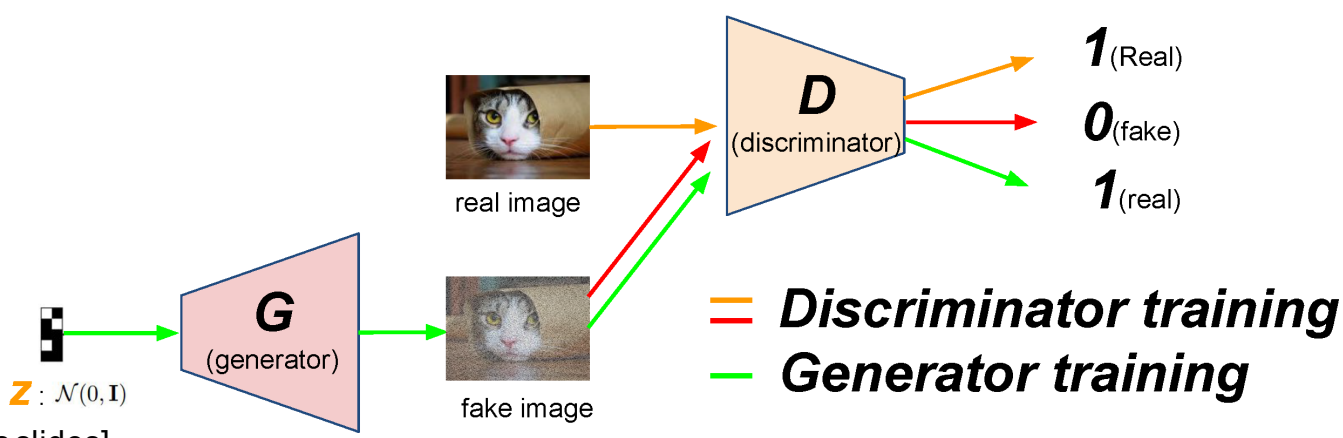
Generative Adversarial Nets (GANs)

- [Goodfellow et al., 2014]
- Assume implicit generative model
- Learn cost function jointly
- Interpreted as a mini-max game between a generator and a discriminator
- Generate sharp, high-fidelity samples

Generative Adversarial Nets (GANs)

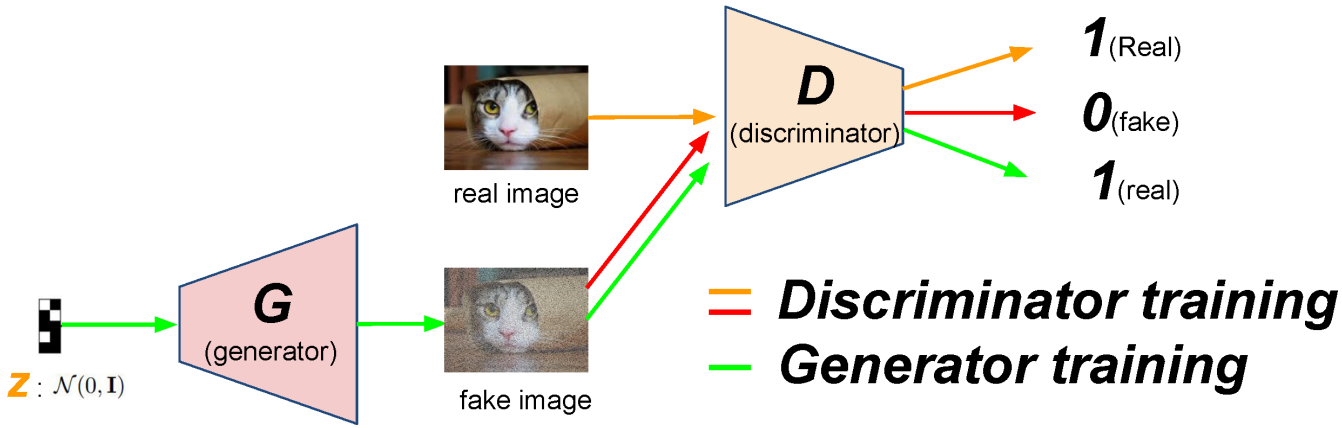
- Generator $\boldsymbol{x}_n = G(\boldsymbol{z}_n, \boldsymbol{\theta}_g) \quad \boldsymbol{z}_n \sim N(\boldsymbol{0}, \boldsymbol{I})$
  - Maps noise variable to data space
- Discriminator $D(\boldsymbol{x}, \boldsymbol{\theta}_d) \in [0, 1]$
  - Outputs the probability that $\boldsymbol{x}$ came from the data rather than the generator

- Learning
  - Train D to maximize the probability of assigning the correct label to both training examples and generated samples
  - Train G to fool the discriminator



[Figure courtesy: Kim's slides]

GANs: Learning



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

- For D: binary cross entropy with label 1 for real, 0 for fake
- For G: $\min_G \mathrm{E}_{\boldsymbol{z}}[\log 1 - D(G(\boldsymbol{z}))]$
- Alternate training of D and G

GANs: Theoretical results

- *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Plug in the minimax objective

$$\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}$$

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

- $C^* = -\log 4$ is the global minimum and the only solution is $p_g = p_{data}$

GANs in Practice

- Optimizing D to completion in inner loop is computational prohibitive
  - Alternate between k steps of optimizing *D* and one step of optimizing *G*

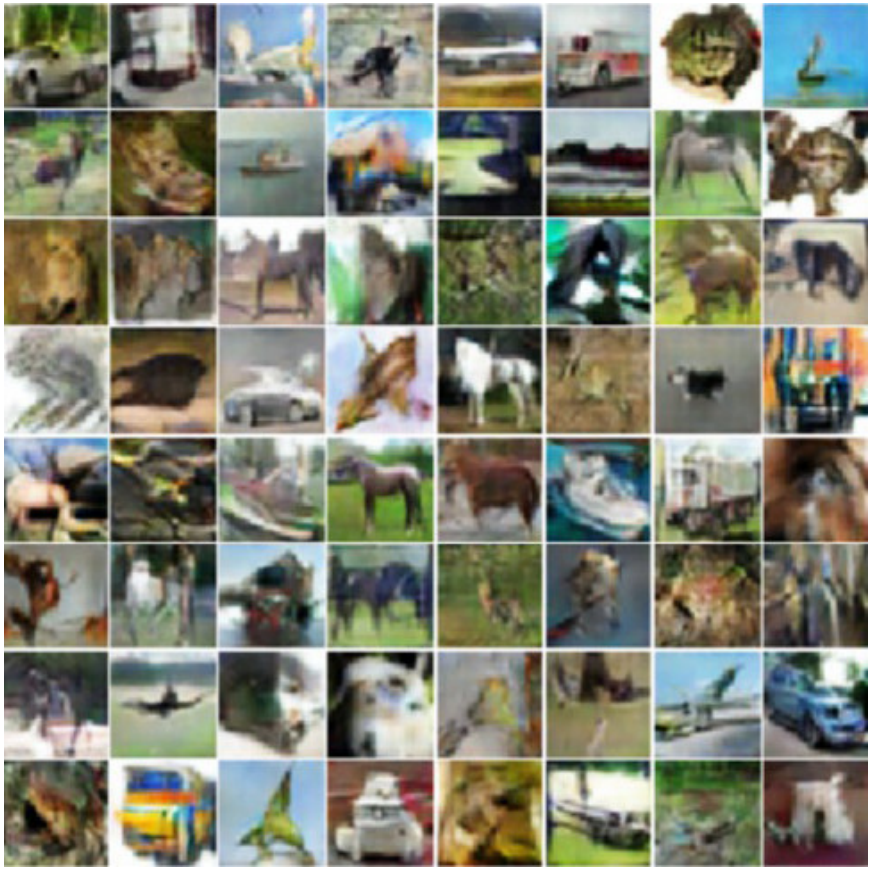
- Optimizing G
  - $\min_G \mathrm{E}_{\boldsymbol{z}}\left[\log 1 - D\left(G(\boldsymbol{z})\right)\right]$ suffers from vanishing gradient when D is too strong
  - Uses $\max_G \mathrm{E}_{\boldsymbol{z}}\left[\log D\left(G(\boldsymbol{z})\right)\right]$ in practice

GANs in Practice

- Instability of training
  - Requires careful balance between the training of D and G
  - [Arjovsky & Botton, 2017]: under certain conditions, $\mathrm{E}_{\boldsymbol{z}}\left[\log D\left(G(\boldsymbol{z})\right)\right]$ is a centered Cauchy distribution with *infinite* expectation and variance
- Mode collapsing
  - Generated samples are often from only a few modes of the data distribution
- A set of heuristics attempting to fix the problems
  - e.g., [Salimans et al., 2016]: minibatch discrimination, one-sided label smoothing, …

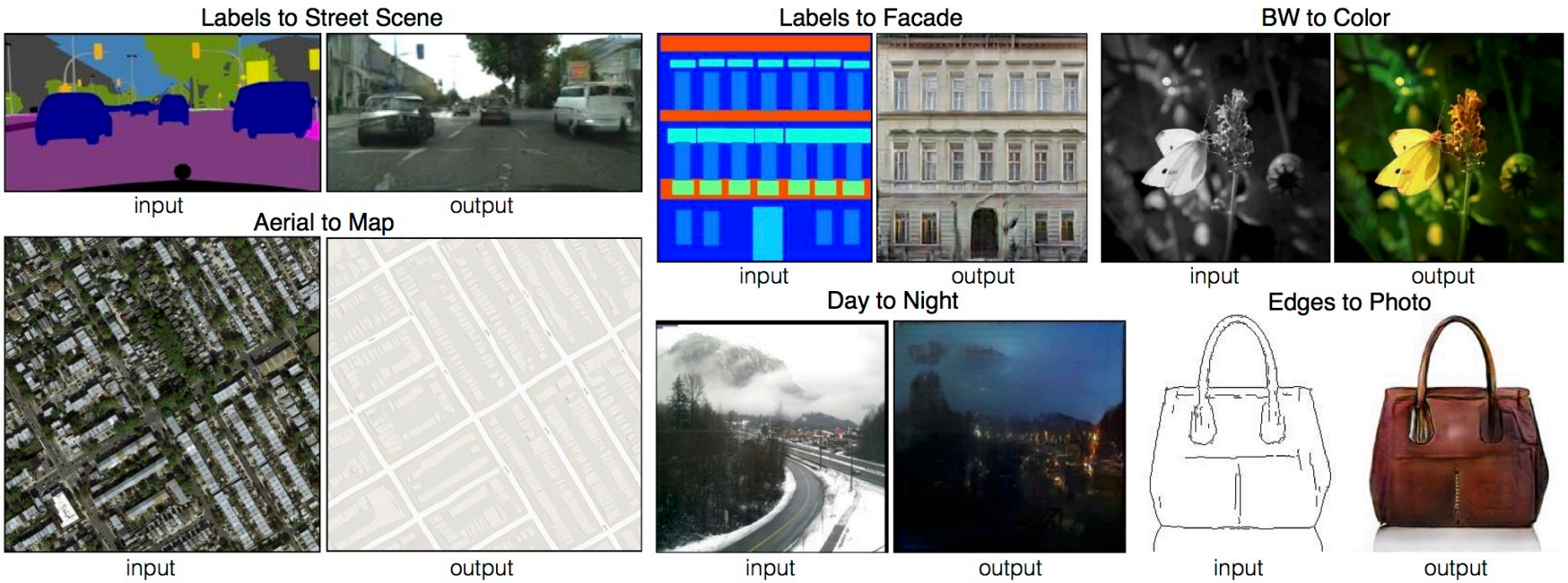
GANs: Applications & results

- Generating images



GANs: Applications & results

- Generating images
- Translating images (e.g., Isola et al., 2016)



GANs: Applications & results

- Generating images
- Translating images (e.g., Isola et al., 2016)
- Domain adaptation (e.g., Purushotham et al., 2017)
- Imitation learning (e.g., Ho & Ermon 2016)
- …

GANs vs. VAEs

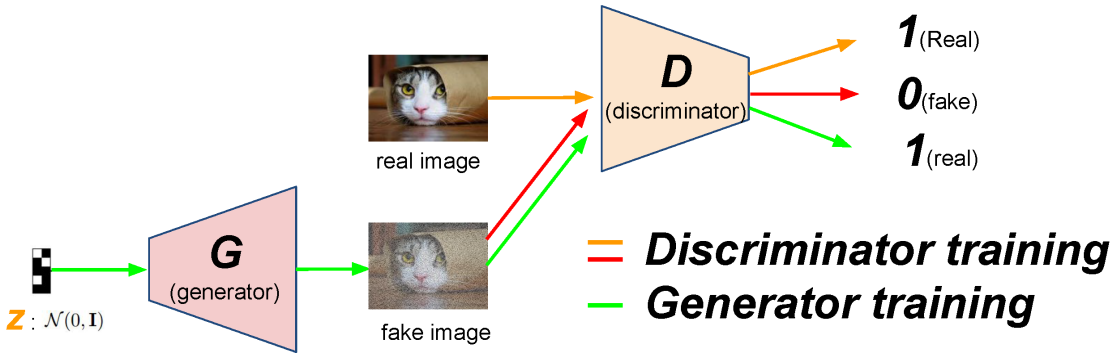
- Variational Auto-encoders:
  - Probabilistic graphical model framework
  - Allow efficient Bayesian inference
  - Generated samples tend to be blurry
    - An issue of maximum likelihood training
  - Do not support discrete latent variables
- GANs:
  - Generate sharp images
  - Do not support inference ($\boldsymbol{x} \rightarrow \boldsymbol{z}$)
  - Do not support discrete visible variables

GANs: Limitations & variants

- Do not support inference
  - No mechanism to inferring $\boldsymbol{z}$ from $\boldsymbol{x}$
  - Variants: additionally learn an inference network
    - [Dumoulin, et al., 2016, Donahue et al., 2016]

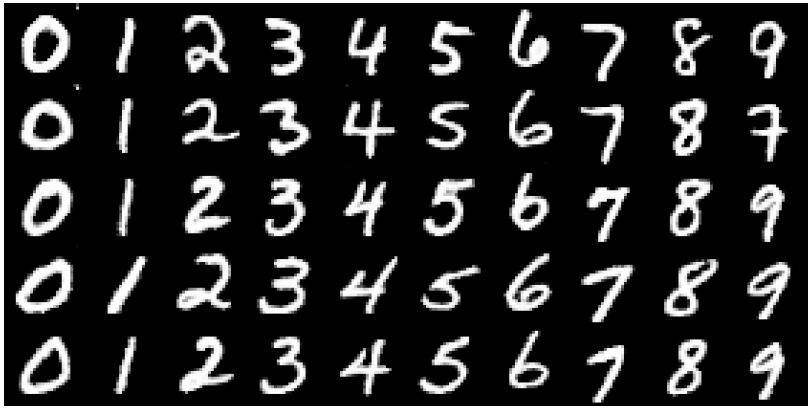
GANs: Limitations & variants

- Do not support discrete visible variables
  - Non-differentiability of samples hinders gradients backprop
  - Variant: treats generator training as policy learning [Yu et al., 2017]
    - High variance, slow convergence
  - Variant: continuous approximations
    - Gumbel-softmax [Kusner & Hernndez-Lobato, 2016]
    - Only qualitative results with small discrete space
  - Variant: combines VAEs with GANs-like discriminators [Hu et al., 2017]
    - Uses VAEs to handle discrete visibles, GANs to handle discrete latents
    - Wake-sleep style learning



GANs: Limitations & variants

- Unable to control the attributes of generated samples
  - Uninterpretability of the input latent vector $\boldsymbol{z}$
  - An Issue shared with VAEs and other DNN methods
  - Variants: add a mutual-information regularizer to enforce disentangled hidden codes [Chen et al., 2016]
    - Unsupervised
    - Semantic of each dimension is observed after training, rather than designated by users in a controllable way



(a) Varying c_1 on InfoGAN (Digit type)

(a) Rotation

GANs: Limitations & variants

- Unable to control the attributes of generated samples
  - Uninterpretability of the input latent vector $\boldsymbol{z}$
  - An Issue shared with VAEs and other DNN methods
  - Variants: add a mutual-information regularizer to enforce disentangled hidden codes [Chen et al., 2016]
    - Unsupervised
    - Semantic of each dimension is observed after training, rather than designated by users in a controllable way
  - Variants: use supervision information to enforce designated semantics on certain dimensions of $\boldsymbol{z}$ [Odena et al., 2017; Hu et al., 2017]

Varying the code of sentiment	Varying the code of tense
this movie was awful and boring .	this was one of the outstanding thrillers of the last decade
this movie was funny and touching .	this is one of the outstanding thrillers of the all time
	this will be one of the great thrillers of the all time
jackson is n't very good with documentary	
jackson is superb as a documentary productions	i thought the movie was too bland and too much
	i guess the movie is too bland and too much
you will regret it	i guess the film will have been too bland
you will enjoy it	

Harnessing DNNs with Logic Rules

- [Hu et al., 2016]
- Deep NNs
  - Heavily rely on massive labeled data
  - Uninterpretable
  - Hard to encode human intention/domain knowledge

Harnessing DNNs with Logic Rules

- How humans learn
  - Learn from *concrete* examples (as DNNs do)
  - Learn from *general* knowledge and rich experiences [Minksy 1980; Lake et al., 2015]
  - E.g., the past tense of verbs[1]:

Examples:

add -> added
accept -> accepted
ignore -> ignored
end -> ended
block -> blocked
love -> loved
…

v.s.

Rule:

regular verbs –d/-ed

[1] https://www.technologyreview.com/s/544606/can-this-man-make-aimore-human

DNNs + knowledge

- Logic rule
  - A flexible declarative language
  - Expresses structured knowledge
  - E.g., sentence sentiment analysis

S has 'A-but-B' structure $\Rightarrow$ (sentiment(S) $\Leftrightarrow$ sentiment(B))

- Input-target space: (X, Y)
- Soft first-order logic (FOL) rules: (r, λ)
  - $r(X, Y) \in [0,1]$
  - λ: confidence level of the rule

Rule knowledge distillation

- Neural network $p_\theta(y|x)$

at iteration t:

true hard label

soft prediction of p_θ

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n))$$

Rule knowledge distillation

- Neural network $p_\theta(y|x)$
- Train to imitate the outputs of a rule-regularized *teacher* network (i.e. distillation)

at iteration t:

true hard label

soft prediction of p_θ

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n))$$

$$\ell(\boldsymbol{s}_n^{(t)}, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)),$$

soft prediction of the teacher network

Rule knowledge distillation

- Neural network $p_\theta(y|x)$
- Train to imitate the outputs of a rule-regularized *teacher* network (i.e. distillation)

at iteration t:

true hard label

soft prediction of p_θ

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{n=1}^{N}(1-\pi)\ell(\boldsymbol{y}_n, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)) + \pi\ell(\boldsymbol{s}_n^{(t)}, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)),$$

balancing parameter

soft prediction of the teacher network

Teacher network construction

- Teacher network: $q(Y|X)$
  - Comes out of p
  - Fits the logic rules: $E_q[r(X,Y)] = 1$, with confidence λ

slack variable

$$\min_{q,\boldsymbol{\xi}\geq 0} \ \mathrm{KL}(q\|p_\theta(\boldsymbol{Y}|\boldsymbol{X})) + C\sum\nolimits_l \xi_l$$

$$\text{s.t.} \ \ \lambda_l(1-\mathbb{E}_q[r_l(\boldsymbol{X},\boldsymbol{Y})]) \leq \xi_l$$

$$l = 1,\ldots,L$$

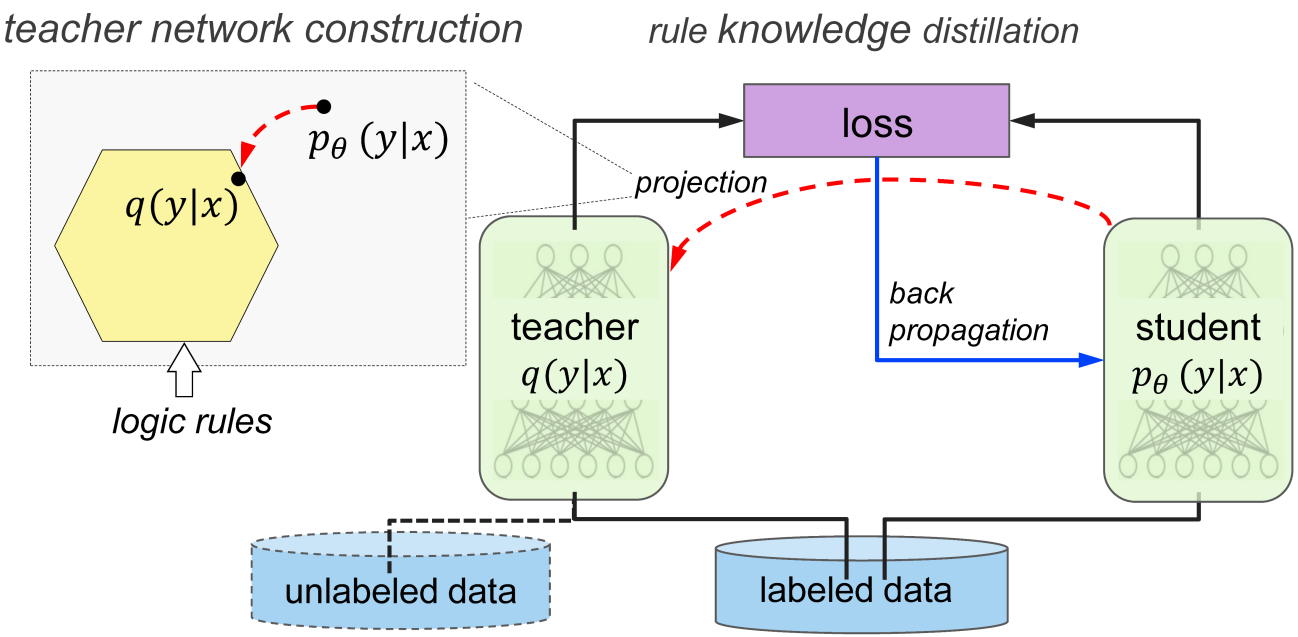
rule constraints

closed-form solution:

$$q^*(\boldsymbol{Y}|\boldsymbol{X}) \propto p_\theta(\boldsymbol{Y}|\boldsymbol{X})\exp\left\{-\sum_l C\lambda_l(1-r_l(\boldsymbol{X},\boldsymbol{Y}))\right\}$$

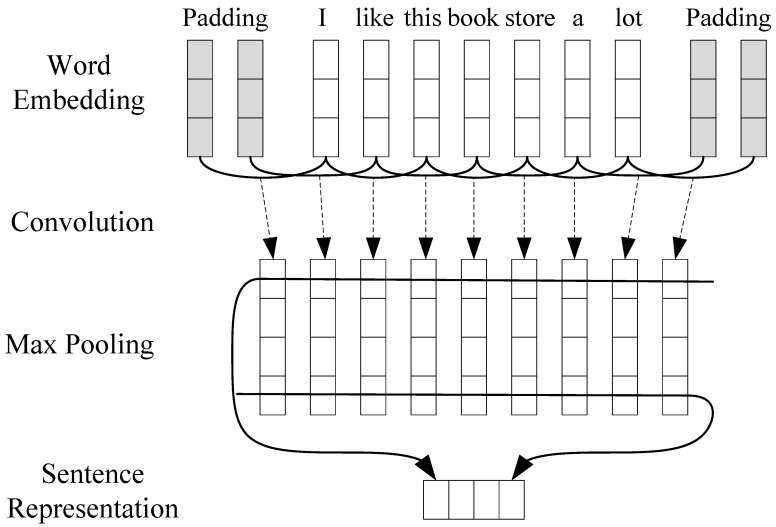
Method summary

- At each iteration
  - Construct a teacher network through posterior constraints
  - Train the NN to emulate the predictions of the teacher



Application: sentiment classification

- Sentence => positive/negative
- Base network: CNN [Kim, 2014]



- Rule knowledge:
  - sentence *S* with structure *A-but-B*:

    => sentiment of *B* dominates

Results

- accuracy (%)

	Model	SST2	MR	CR
1	CNN (Kim, 2014)	87.2	81.3±0.1	84.3±0.2
2	CNN-Rule-p	88.8	81.6±0.1	85.0±0.3
3	CNN-Rule-q	89.3	**81.7±0.1**	**85.3±0.3**
4	MGNC-CNN (Zhang et al., 2016)	88.4	–	–
5	MVCNN (Yin and Schutze, 2015)	**89.4**	–	–
6	CNN-multichannel (Kim, 2014)	88.1	81.1	85.0
7	Paragraph-Vec (Le and Mikolov, 2014)	87.8	–	–
8	CRF-PR (Yang and Cardie, 2014)	–	–	82.7
9	RNTN (Socher et al., 2013)	85.4	–	–
10	G-Dropout (Wang and Manning, 2013)	–	79.0	82.1

References

- [Dosovitskiy et al., 2016] “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, NIPS’16
- [Larsen et al., 2016] “Autoencoding beyond pixels using a learned similarity metric”, ICML’16
- [Kingma et al., 2014] “Semi-supervised learning with deep generative models”, NIPS’14
- [Jang et al., 2017] “Categorical Reparameterization with Gumbel-Softmax”, ICLR’17
- [Hu et al., 2017] “Controllable Text Generation”, 2017
- [Goyal et al., 2017] “Nonparametric Variational Auto-encoders for Hierarchical Representation Learning”, 2017
- [Goodfellow et al., 2014] “Generative Adversarial Nets”, NIPS’14
- [Arjovsky & Botton, 2017] “Towards Principled Methods for Training Generative Adversarial Networks”, ICLR’17
- [Salimans et al., 2016] “Improved Techniques for Training GANs”, NIPS’16
- [Isola et al., 2016] “Image-to-Image Translation with Conditional Adversarial Networks”, 2016
- [Purushotham et al., 2017] “Variational Recurrent Adversarial Deep Domain Adaptation”, ICLR’17
- [Ho & Ermon 2016] “Generative Adversarial Imitation Learning”, NIPS’16
- [Dumoulin, et al., 2016] “Adversarially Learned Inference”, ICLR’17
- [Donahue et al., 2016] “Adversarial Feature Learning”, 2016
- [Yu et al., 2017] “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”, AAAI’17
- [Kusner & Hernndez-Lobato, 2016] “GANs for sequences of discrete elements with the gumbel-softmax distribution”, 2016
- [Chen el al., 2016] “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”, NIPS’16
- [Odena et al., 2017] “Conditional image synthesis with auxiliary classifier GANs”, 2017
- [Hu et al., 2016] “Harnessing DNNs with logic rules”, ACL’2016
- [Minksy, 1980] Learning meaning. Technical Report AI Lab Memo. 1980
- [Lake et al., 2015] Human-level concept learning through probabilistic program induction. Science’15.
- [Kim, 2014] Convolutional neural networks for sentence classification. EMNLP’14
- Hamid Reza Maei, Wake-Sleep algorithm for Representational Learning
- Mark Chang, Variational Autoencoder
- Namju Kim, Generative Adversarial Networks (GAN)