PETUUM

Carnegie Mellon University
School of Computer Science

Probabilistic Graphical Models

Deep Generative Models - II

Eric Xing

Lecture 13, February 26, 2020

Reading: see class homepage

Outline

- Generative Adversarial Networks (GANs)
  - GANs Progress
  - Vanilla GAN, Wasserstein GAN, Progressive GAN, BigGAN
- Normalizing Flow (NF)
  - Basic Concepts
  - GLOW
- Integrating Domain Knowledge into Deep Learning

Outline

- Generative Adversarial Networks (GANs)
  - GANs Progress
  - Vanilla GAN, Wasserstein GAN, Progressive GAN, BigGAN
- Normalizing Flow (NF)
  - Basic Concepts
  - GLOW
- Integrating Domain Knowledge into Deep Learning

GAN Progress on Face Generation

Figure courtesy: Ian Goodfellow

Recap: Generative Adversarial Nets (GANs)

- Generative model $\boldsymbol{x} = G_\theta(\boldsymbol{z}), \ \ \boldsymbol{z} \sim p(\boldsymbol{z})$
  - Map noise variable $\boldsymbol{z}$ to data space $\boldsymbol{x}$
  - Define an implicit distribution over $\boldsymbol{x}$: $p_{g_\theta}(\boldsymbol{x})$
    - a stochastic process to simulate data $\boldsymbol{x}$
    - Intractable to evaluate likelihood
- Discriminator $D_\phi(\boldsymbol{x})$
  - Output the probability that $\boldsymbol{x}$ came from the data rather than the generator

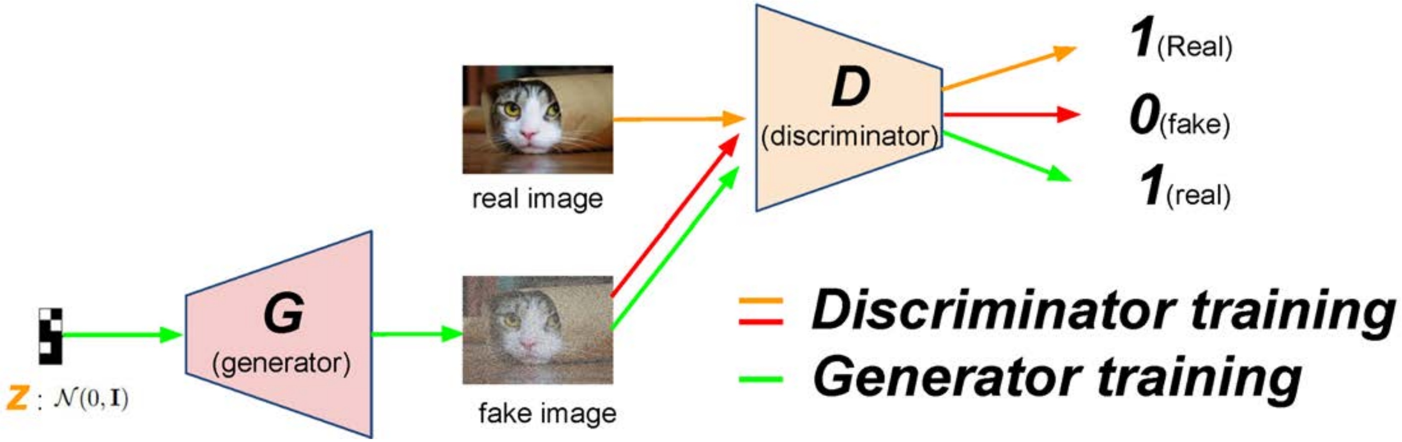


Figure courtesy: Kim

Recap: Generative Adversarial Nets (GANs)

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

$$\max_D \mathcal{L}_D = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - D(\boldsymbol{x}))\right]$$
$$\min_G \mathcal{L}_G = \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - D(\boldsymbol{x}))\right].$$

- [Goodfellow et al., 2014]

  $\min_\theta \text{JSD}(\, P_{data} \,||\, P_{g_\theta} \,)$

- [Hu et al., 2017]

  $\min_\theta \text{KL}(\, P_\theta \,||\, Q)$

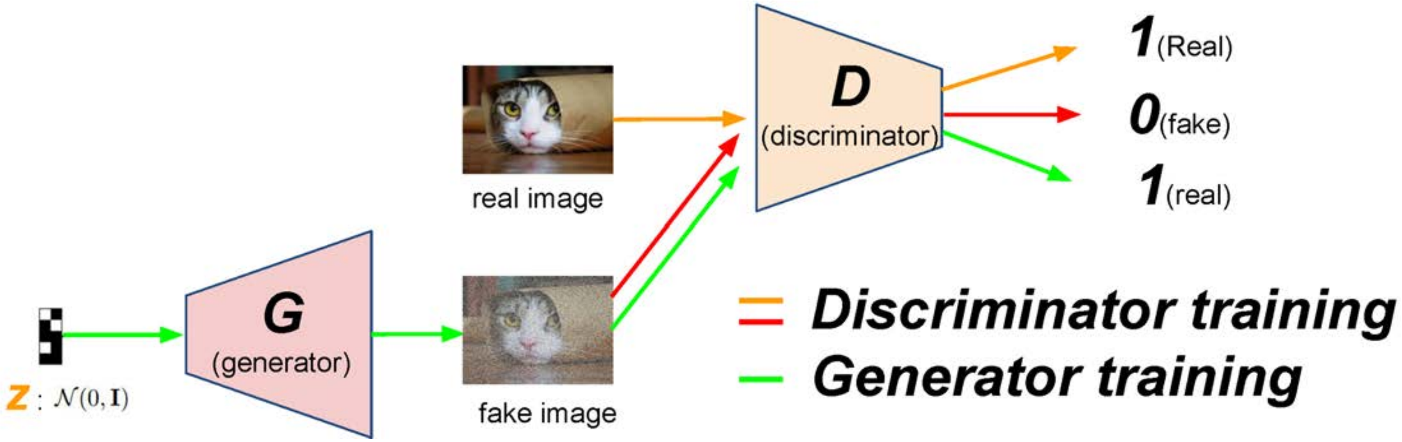


Figure courtesy: Kim

Wasserstein GAN (WGAN)

- If our data are on a low-dimensional manifold of a high dimensional space, the model's manifold and the true data manifold can have a negligible intersection in practice

[Arjovsky et al., 2017] Slide adapted from bhiksha

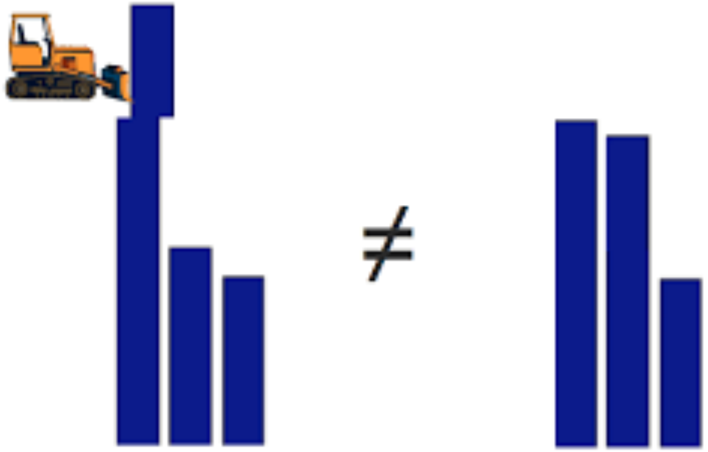
Wasserstein GAN (WGAN)

- If our data are on a low-dimensional manifold of a high dimensional space, the model's manifold and the true data manifold can have a negligible intersection in practice
- KL divergence is undefined or infinite
- The loss function and gradients may not be continuous and well behaved

[Arjovsky et al., 2017] Slide adapted from bhiksha

Wasserstein GAN (WGAN)

- If our data are on a low-dimensional manifold of a high dimensional space, the model's manifold and the true data manifold can have a negligible intersection in practice
- KL divergence is undefined or infinite
- The loss function and gradients may not be continuous and well behaved
- The Wasserstein Distance is well defined
  - Earth Mover's Distance
  - Minimum transportation cost for making one pile of dirt in the shape of one probability distribution to the shape of the other distribution



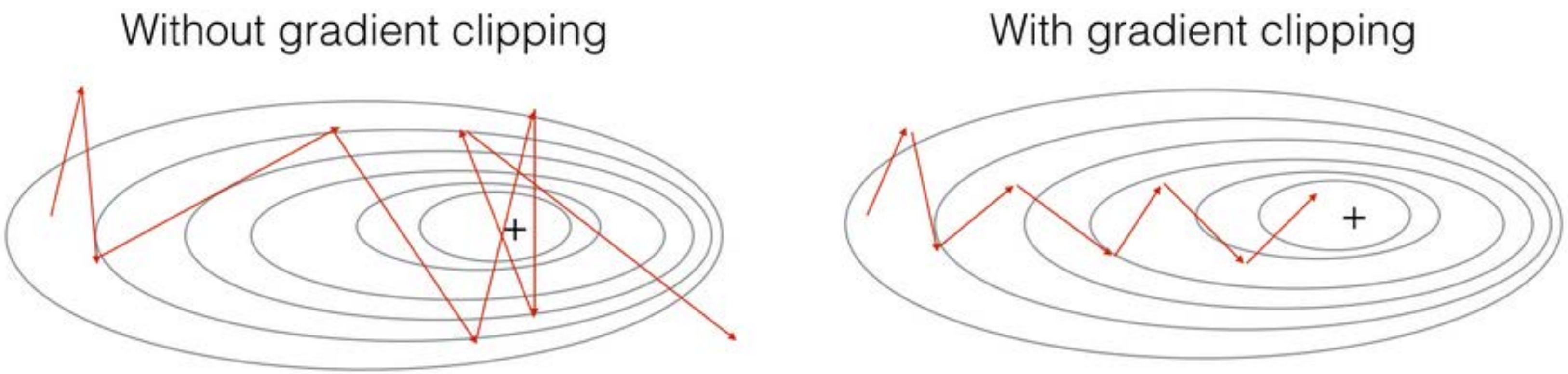
[Arjovsky et al., 2017] Slide adapted from bhiksha

Wasserstein GAN (WGAN)

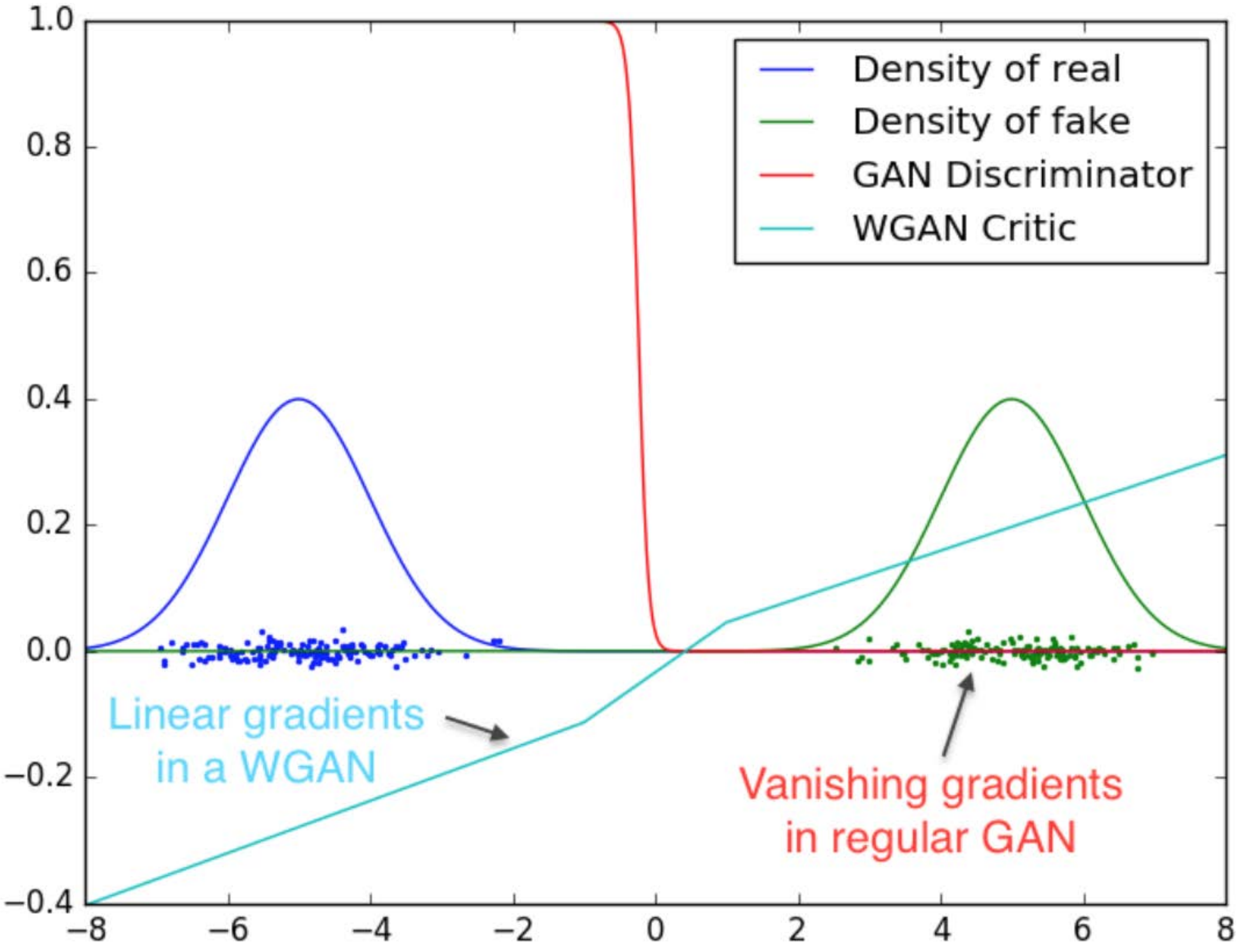
- Objective

$$W(p_{data}, p_g) = \frac{1}{K} \sup_{||D||_L \leq K} \mathrm{E}_{x \sim p_{data}}[D(x)] - \mathrm{E}_{x \sim p_g}[D(x)]$$

- $||D||_L \leq K$: K- Lipschitz continuous
- Use gradient-clipping to ensure D has the Lipschitz continuity



WGAN vs Vanilla GAN



Progressive GAN

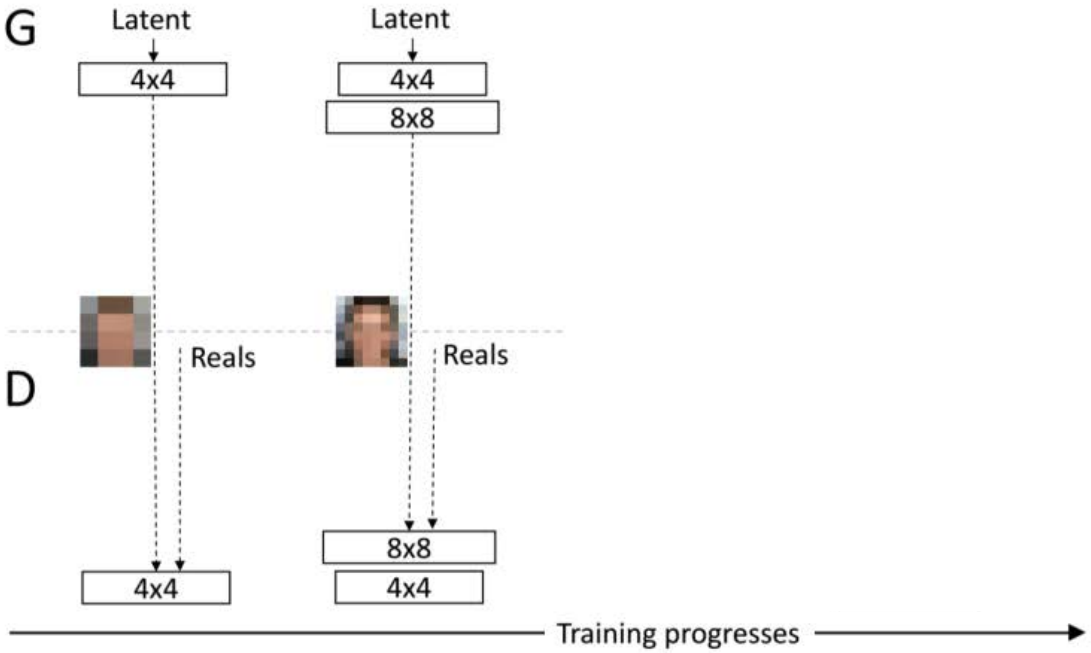
Low resolution images

[Karras et al., 2018]

Progressive GAN

Low resolution images

add in additional layers



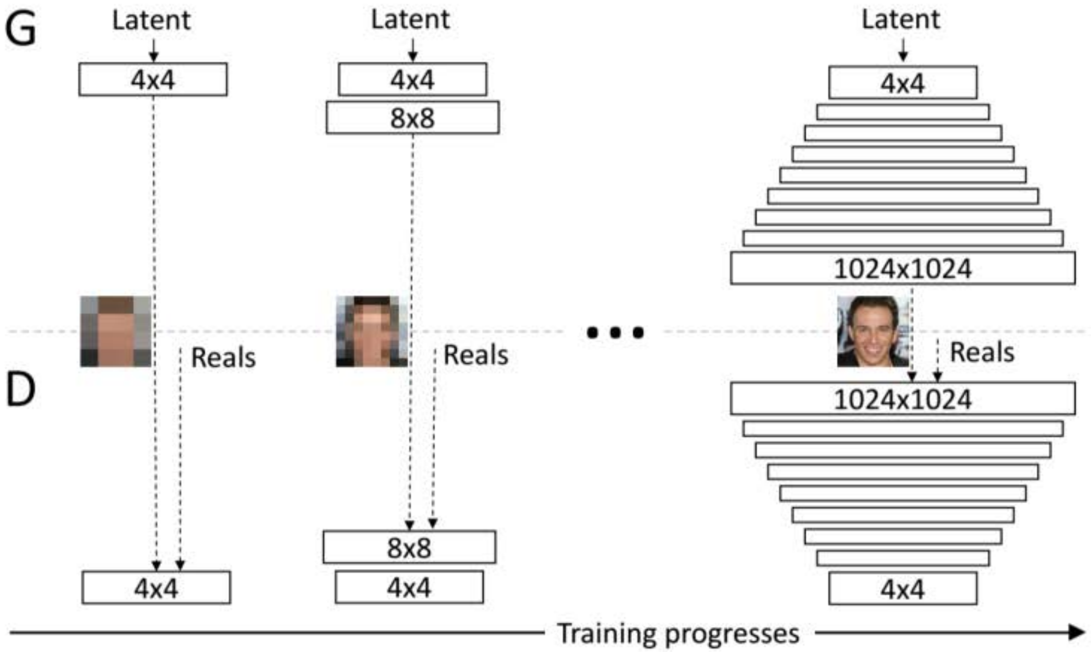
[Karras et al., 2018]

Progressive GAN

Low resolution images

add in additional layers

High resolution images



[Karras et al., 2018]

BigGAN

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling
- 2x – 4x more parameters
- 8x larger batch size
- Simple architecture changes that improve scalability

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling
- 2x – 4x more parameters
- 8x larger batch size
- Simple architecture changes that improve scalability

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling
- 2x
- 8x
- Sir

[Brock et al., 2018]

Outline

- Generative Adversarial Networks (GANs)
  - GANs Progress
  - Vanilla GAN, Wasserstein GAN, Progressive GAN, BigGAN

- Normalizing Flow (NF)
  - Basic Concepts
  - GLOW

- Integrating Domain Knowledge into Deep Learning

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

Figure courtesy: Lilian Weng

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

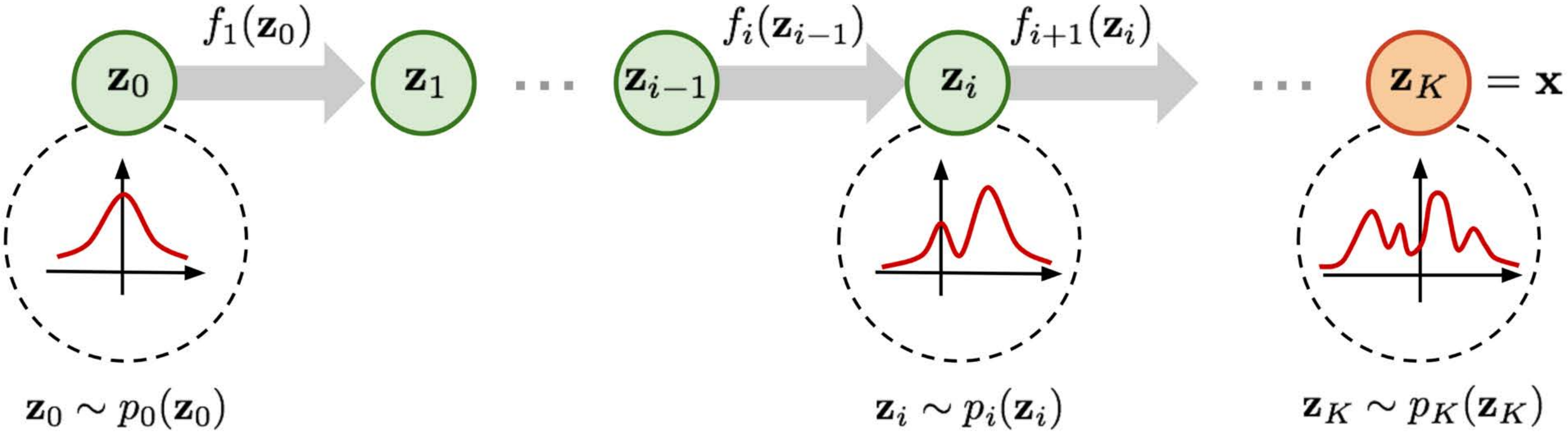
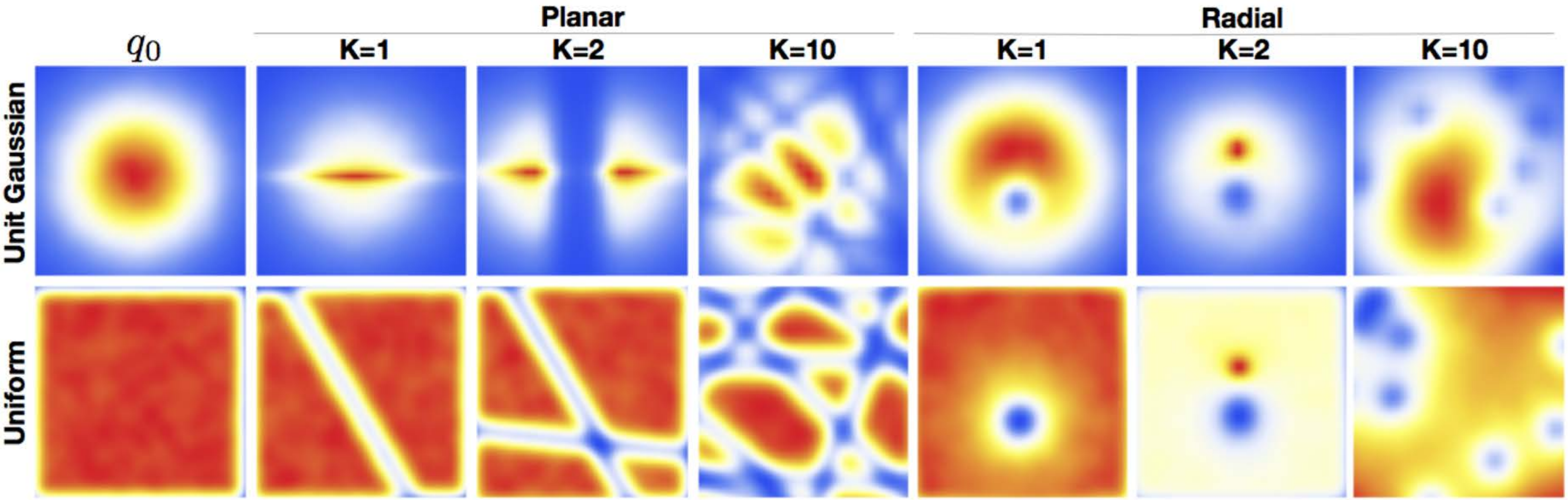


Figure courtesy: Lilian Weng

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions



[Rezende & Mohamed, 2015]

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

inference: $\boldsymbol{z} = f^{-1}(\boldsymbol{x})$

Transformation function f

- Invertible

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$z \sim p(z)$$
$$x = f(z)$$

inference: $z = f^{-1}(x)$

density: $p(x) = p(z) \left|\det \frac{dz}{dx}\right|$

$$= p(f^{-1}(x)) \left|\det \frac{df^{-1}}{dx}\right|$$

$\det \frac{df^{-1}}{dx}$ -- Jacobian determinant

Transformation function f

- Invertible

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

Transformation function f

inference: $\boldsymbol{z} = f^{-1}(\boldsymbol{x})$ - - - - -> • Invertible

density: $p(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \det \frac{d\boldsymbol{z}}{d\boldsymbol{x}} \right|$ - - - - -> • Jacobian determinant easy to compute

e.g., choose $df^{-1}/d\boldsymbol{x}$ to be a triangular matrix

$$= p(f^{-1}(\boldsymbol{x})) \left| \det \frac{df^{-1}}{d\boldsymbol{x}} \right|$$

$\det \frac{df^{-1}}{d\boldsymbol{x}}$ -- Jacobian determinant

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0)$$
$$\boldsymbol{x} = \boldsymbol{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\boldsymbol{z}_0)$$

inference: $\boldsymbol{z}_i = f_i^{-1}(\boldsymbol{z}_{i-1})$

density: $p(\boldsymbol{z}_i) = p(\boldsymbol{z}_{i-1}) \left|\det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i}\right|$

Transformation function f_i

- Invertible
- Jacobian determinant easy to compute
  e.g., choose $df_i^{-1}/d\boldsymbol{z}_i$ to be a triangular matrix

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0)$$
$$\boldsymbol{x} = \boldsymbol{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\boldsymbol{z}_0)$$

inference: $\boldsymbol{z}_i = f_i^{-1}(\boldsymbol{z}_{i-1})$

density: $p(\boldsymbol{z}_i) = p(\boldsymbol{z}_{i-1}) \left| \det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i} \right|$

Transformation function f_i

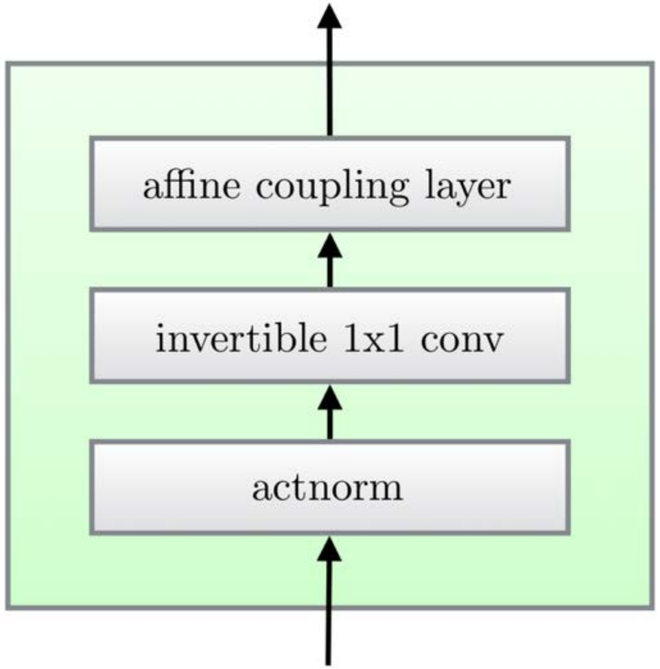
- Invertible
- Jacobian determinant easy to compute
  e.g., choose $df_i^{-1}/d\boldsymbol{z}_i$ to be a triangular matrix

training: maximizes data log-likelihood

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}_0) + \sum_{i=1}^{K} \log \left| \det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i} \right|$$

GLOW

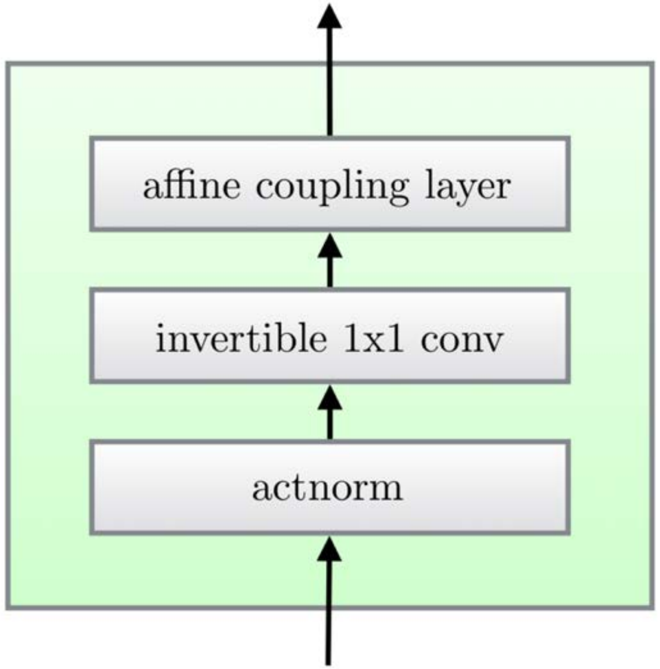
- [Kingma and Dhariwal., 2018]



One step of flow in the Glow model

GLOW

- [Kingma and Dhariwal., 2018]



One step of flow in the Glow model

Outline

- Generative Adversarial Networks (GANs)
  - GANs Progress
  - Vanilla GAN, Wasserstein GAN, Progressive GAN, BigGAN
- Normalizing Flow (NF)
  - Basic Concepts
  - GLOW
- Integrating Domain Knowledge into Deep Learning

Deep Learning

- Heavily rely on massive labeled data

Deep Learning

- Heavily rely on massive labeled data
- Uninterpretable

Deep Learning

- Heavily rely on massive labeled data
- Uninterpretable
- Hard to encode human intention and domain knowledge

How Humans Learn

- Learn from concrete examples (as DNNs do)
- Learn from abstract knowledge (definitions, logic rules, etc) [Minksy 1980; Lake et al., 2015]

How Humans Learn

- Learn from **concrete** examples (as DNNs do)
- Learn from **abstract** knowledge (definitions, logic rules, etc) [Minksy 1980; Lake et al., 2015]

Past tense of verb

Examples:

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

V.S.

Rule:

regular verbs –d/-ed

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

Integrating Domain Knowledge into Deep Learning

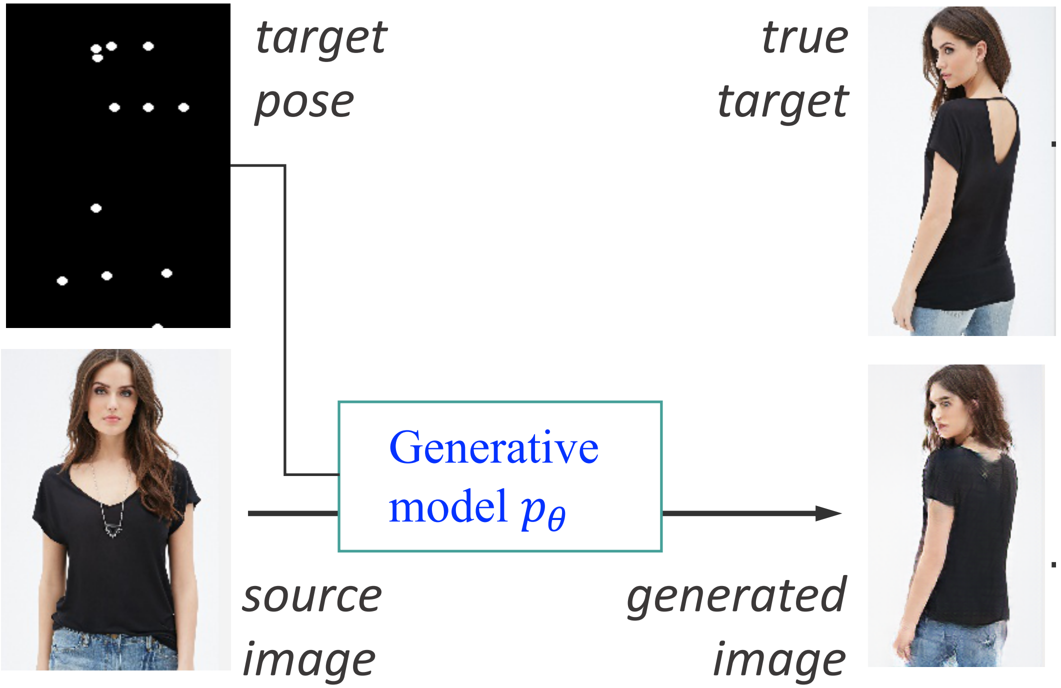
- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
  - Conditional model, $p_\theta(\boldsymbol{x}|\ inputs)$
  - Generative model, e.g., $\boldsymbol{x}$ is an image
  - Discriminative model, e.g., x is a sentence label

Integrating Domain Knowledge into Deep Learning

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$
  - Higher f_ϕ value, better $\boldsymbol{x}$ w.r.t. the knowledge

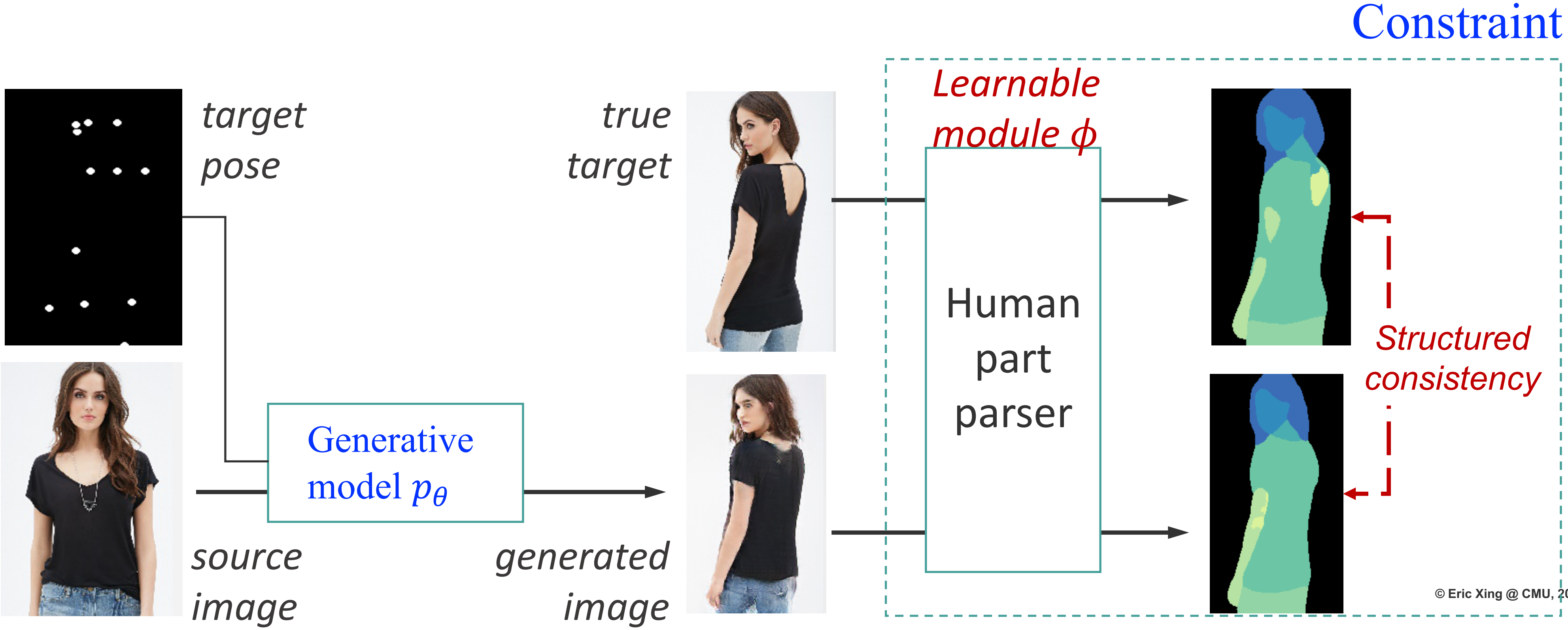
Integrating Domain Knowledge into Deep Learning

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$
  - Higher f_ϕ value, better $\boldsymbol{x}$ w.r.t. the knowledge



Integrating Domain Knowledge into Deep Learning

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$
  - Higher f_ϕ value, better $\boldsymbol{x}$ w.r.t. the knowledge



Integrating Domain Knowledge into Deep Learning

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$
  - Higher f_ϕ value, better $\boldsymbol{x}$ w.r.t. the knowledge

- Sentiment classification
  - "This was a terrific movie, but the director could have done better"

- Logical Rules:
  - Sentence S with structure A-but-B => sentiment of *B* dominates

Learning with Constraints

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$
  - Higher f_ϕ value, better $\boldsymbol{x}$ w.r.t. the knowledge

- One way to impose the constraint is to maximize: $\mathbb{E}_{p_\theta}[f_\phi(\boldsymbol{x})]$

Learning with Constraints

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$
  - Higher f_ϕ value, better $\boldsymbol{x}$ w.r.t. the knowledge

- One way to impose the constraint is to maximize: $\mathbb{E}_{p_\theta}[f_\phi(\boldsymbol{x})]$
- Objective:

$$\min_\theta \ \mathcal{L}(\boldsymbol{\theta}) - \alpha \, \mathbb{E}_{p_\theta}\left[f_\phi(\boldsymbol{x})\right]$$

Regular objective (e.g., cross-entropy loss, etc.)

Regularization: imposing constraints (difficult to compute)

Learning with Constraints

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$

$$\min_\theta \ \mathcal{L}(\boldsymbol{\theta}) - \alpha \, \mathbb{E}_{p_\theta}\left[f_\phi(\boldsymbol{x})\right]$$

Learning with Constraints

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$

$$\min_\theta \; \mathcal{L}(\boldsymbol{\theta}) - \alpha \, \mathbb{E}_{p_\theta}\left[f_\phi(\boldsymbol{x})\right]$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})) - \lambda \, \mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

Slide courtesy: Ruslan Salakhutdinov

Learning with Constraints

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$

$$\min_\theta \ \mathcal{L}(\boldsymbol{\theta}) - \alpha \, \mathbb{E}_{p_\theta}\left[f_\phi(\boldsymbol{x})\right]$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})) - \lambda \, \mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

Posterior Regularization
[Ganchev et al., 2010]

- Introduce variational distribution q
  - Impose constraint on q
  - Encourage q to stay close to p

Learning with Constraints

- Consider a statistical model $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$
- Consider a constraint function $f_\phi(\boldsymbol{x}) \in \mathbb{R}$

$$\min_\theta \; \mathcal{L}(\boldsymbol{\theta}) - \alpha \, \mathbb{E}_{p_\theta}\left[f_\phi(\boldsymbol{x})\right]$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{x})) - \lambda \, \mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

Posterior Regularization
[Ganchev et al., 2010]

- Introduce variational distribution q
  - Impose constraint on q
  - Encourage q to stay close to p
- Objective

$$\min_{\theta, q} \; \mathcal{L}(\boldsymbol{\theta}) + \alpha \, \mathcal{L}(\boldsymbol{\theta}, q)$$

Learning with Constraints

$$\min_{\theta,q}\ \mathcal{L}(\boldsymbol{\theta}) + \alpha\, \mathcal{L}(\boldsymbol{\theta}, q)$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x}) || \, p_\theta(\boldsymbol{x})) - \lambda\, \mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

Learning with Constraints

$$\min_{\theta,q} \ \mathcal{L}(\boldsymbol{\theta}) + \alpha \, \mathcal{L}(\boldsymbol{\theta}, q)$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x}) || \, p_\theta(\boldsymbol{x})) - \lambda \, \mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

- EM algorithm for solving the problem
  - E-step

$$q^*(\boldsymbol{x}) = p_\theta(\boldsymbol{x}) \exp\left\{ \lambda f_\phi(\boldsymbol{x}) \right\}/Z$$

Learning with Constraints

$$\min_{\theta,q}\ \mathcal{L}(\boldsymbol{\theta}) + \alpha\,\mathcal{L}(\boldsymbol{\theta}, q)$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x}) || \, p_\theta(\boldsymbol{x})) - \lambda\, \mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

- EM algorithm for solving the problem
  - E-step

$$q^*(\boldsymbol{x}) = p_\theta(\boldsymbol{x}) \exp\{\,\lambda f_\phi(\boldsymbol{x})\,\}/Z$$

Higher value -- higher probability under q – "soft constraint"

Learning with Constraints

$$\min_{\theta,q}\ \mathcal{L}(\boldsymbol{\theta}) + \alpha\,\mathcal{L}(\boldsymbol{\theta}, q)$$

$$\mathcal{L}(\boldsymbol{\theta}, q) = \mathrm{KL}(q(\boldsymbol{x})\,||\,p_\theta(\boldsymbol{x})) - \lambda\,\mathbb{E}_q\left[f_\phi(\boldsymbol{x})\right]$$

- EM algorithm for solving the problem
  - E-step

$$q^*(\boldsymbol{x}) = p_\theta(\boldsymbol{x})\exp\{\,\lambda f_\phi(\boldsymbol{x})\,\}/Z$$

Higher value -- higher probability under q – "soft constraint"

  - M-step

$$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(\boldsymbol{x})]$$

Logical Rule Constraints

- Consider a supervised learning: $p_\theta(y|\boldsymbol{x})$

[Hu et al., 2016]

Logical Rule Constraints

- Consider a supervised learning: $p_\theta(y|\boldsymbol{x})$
- Input-Target space (X, Y)

[Hu et al., 2016]

Logical Rule Constraints

- Consider a supervised learning: $p_\theta(y|\boldsymbol{x})$
- Input-Target space (X, Y)
- First-order logic rules: (r, λ)
    - $r(X, Y) \in [0, 1]$, could be soft
    - λ is the confidence level of the rule

[Hu et al., 2016]

Logical Rule Constraints

- Consider a supervised learning: $p_\theta(y|\boldsymbol{x})$
- Input-Target space (X, Y)
- First-order logic rules: (r, λ)
  - $r(X, Y) \in [0, 1]$, could be soft
  - λ is the confidence level of the rule

- Given l rules:
  - E-step:
$$q^*(y|\boldsymbol{x}) = p_\theta(y|\boldsymbol{x}) \exp\left\{\sum_l \lambda_l r_l(y, \boldsymbol{x})\right\} / Z$$
  - M-step:
$$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(y|\boldsymbol{x})]$$

[Hu et al., 2016]

Logical Rule Constraints

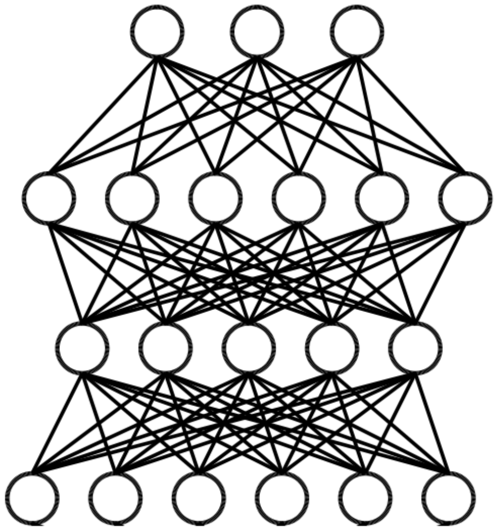
- Consider a supervised learning: $p_\theta(y|\boldsymbol{x})$
- Input-Target space (X, Y)
- First-order logic rules: (r, λ)
  - $r(X, Y) \in [0, 1]$, could be soft
  - λ is the confidence level of the rule
- Given l rules:
  - E-step: $$q^*(y|\boldsymbol{x}) = p_\theta(y|\boldsymbol{x}) \exp\left\{\sum_l \lambda_l r_l(y, \boldsymbol{x})\right\} / Z$$
  - M-step: $$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(y|\boldsymbol{x})]$$

Knowledge distillation [Hinton et al., 2015; Bucilu et al., 2006]

[Hu et al., 2016]

Knowledge Distillation

$p_\theta(y|\boldsymbol{x})$

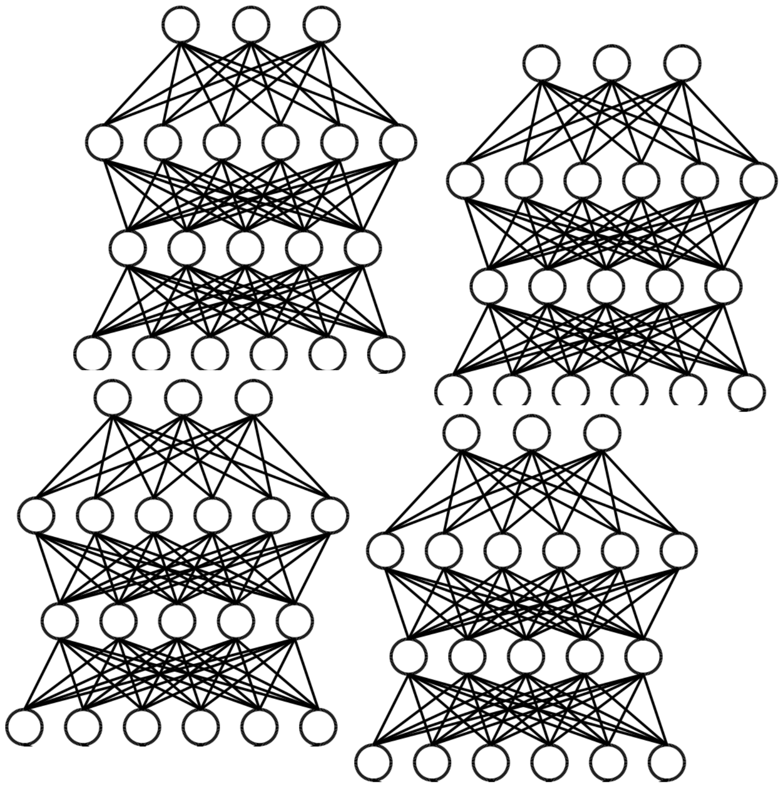


Student

[Hinton et al., 2015; Bucilu et al., 2006]

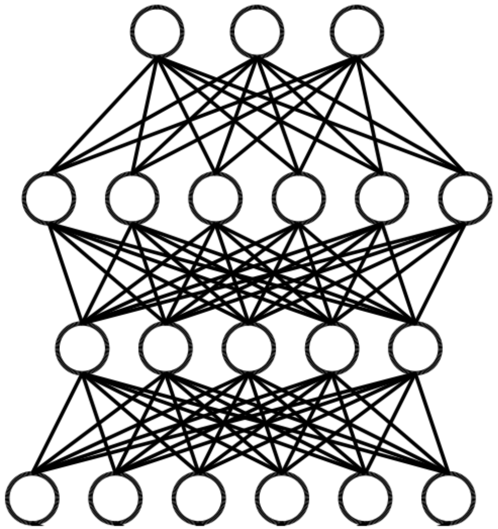
Knowledge Distillation

$q(y|\boldsymbol{x})$



Teacher
(Ensemble)

$p_\theta(y|\boldsymbol{x})$



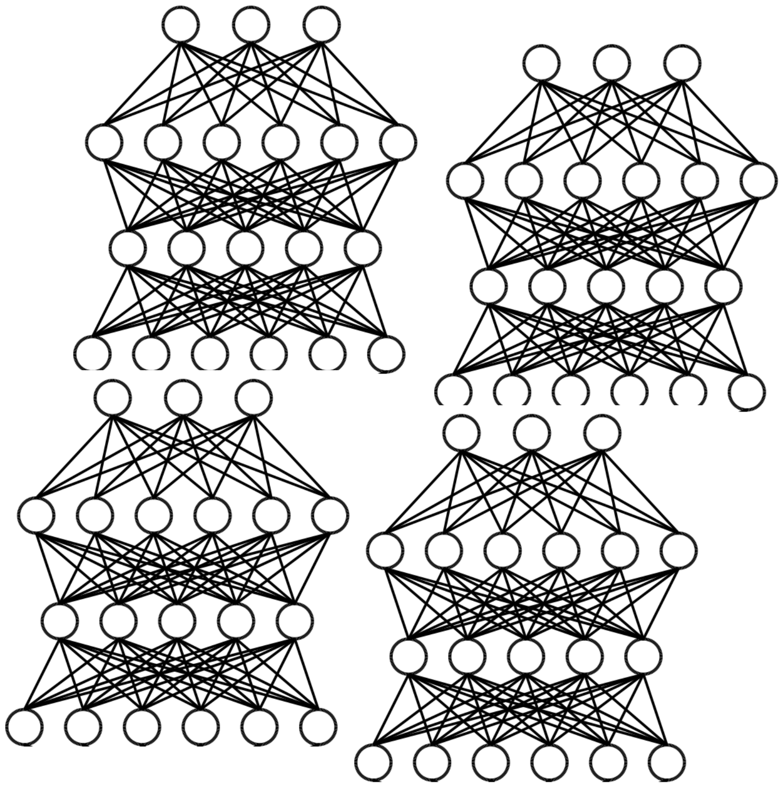
Student

[Hinton et al., 2015; Bucilu et al., 2006]

Knowledge Distillation

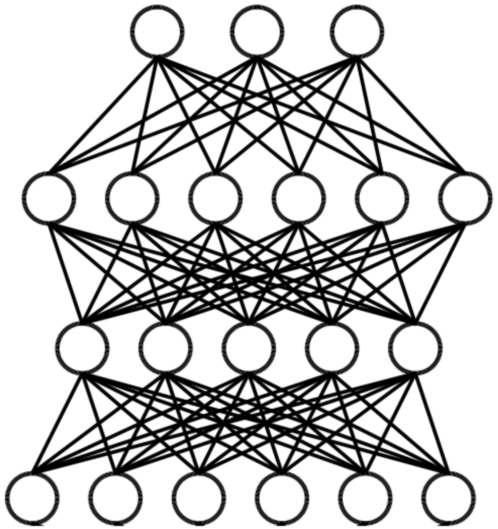
Match soft predictions of the teacher network and student network

$q(y|\boldsymbol{x})$



Teacher
(Ensemble)

$p_\theta(y|\boldsymbol{x})$



Student

[Hinton et al., 2015; Bucilu et al., 2006]

Rule Knowledge Distillation

$$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(y|\boldsymbol{x})]$$

- Neural network $p_\theta(y|\boldsymbol{x})$
- Train to imitate the outputs of the rule-regularized teacher network

[Hu et al., 2016]

Rule Knowledge Distillation

$$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(y|\boldsymbol{x})]$$

- Neural network $p_\theta(y|\boldsymbol{x})$
- Train to imitate the outputs of the rule-regularized teacher network
- At iteration t:

true hard label — soft prediction of $p_\theta(y|x)$

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

[Hu et al., 2016]

Rule Knowledge Distillation

$$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(y|\boldsymbol{x})]$$

- Neural network $p_\theta(y|\boldsymbol{x})$
- Train to imitate the outputs of the rule-regularized teacher network
- At iteration t:

true hard label

soft prediction of $p_\theta(y|x)$

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

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

soft prediction of the teacher network

$$q^*(y|\boldsymbol{x}) = p_\theta(y|\boldsymbol{x}) \exp\left\{\sum_l \lambda_l r_l(y, \boldsymbol{x})\right\}/Z$$

[Hu et al., 2016]

Rule Knowledge Distillation

$$\min_\theta \mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}_{q^*}[\log p_\theta(y|\boldsymbol{x})]$$

- Neural network $p_\theta(y|\boldsymbol{x})$
- Train to imitate the outputs of the rule-regularized teacher network
- At iteration t:

true hard label

soft prediction of $p_\theta(y|x)$

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

soft prediction of the teacher network

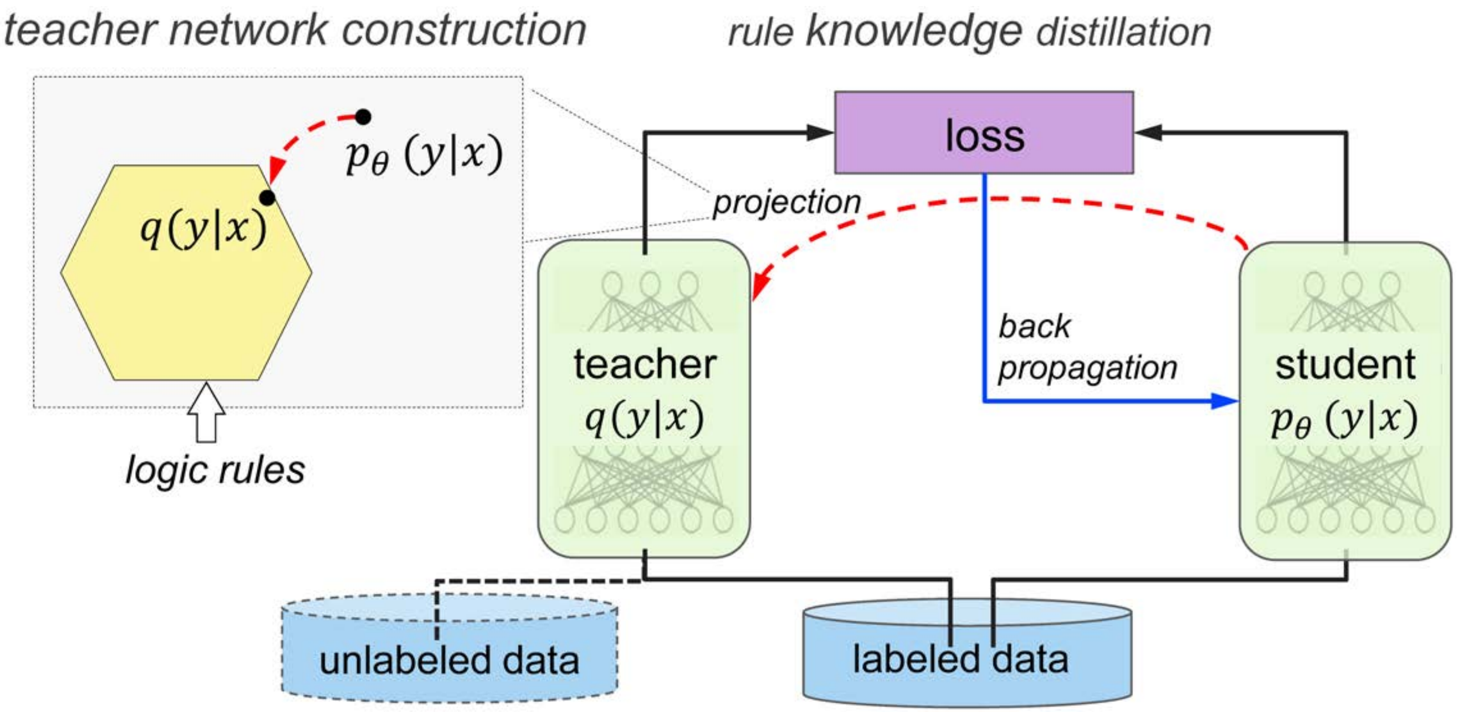
balancing parameter

$$q^*(y|\boldsymbol{x}) = p_\theta(y|\boldsymbol{x}) \exp\left\{\sum_l \lambda_l r_l(y, \boldsymbol{x})\right\} / Z$$

[Hu et al., 2016]

Rule Knowledge Distillation

- Neural network $p_\theta(y|\boldsymbol{x})$
- At each iteration
  - Construct a teacher network with "soft constraint"
  - Train DNN to emulate the teacher network



Learning Rules / Constraints

$$q^*(y|\boldsymbol{x}) = p_\theta(y|\boldsymbol{x}) \exp\left\{\sum_l \lambda_l r_l(y, \boldsymbol{x})\right\} / Z$$

- Learn the confidence value λ_l for each logical rule [Hu et al., 2016b]

Slide adapted from Ruslan Salakhutdinov

Learning Rules / Constraints

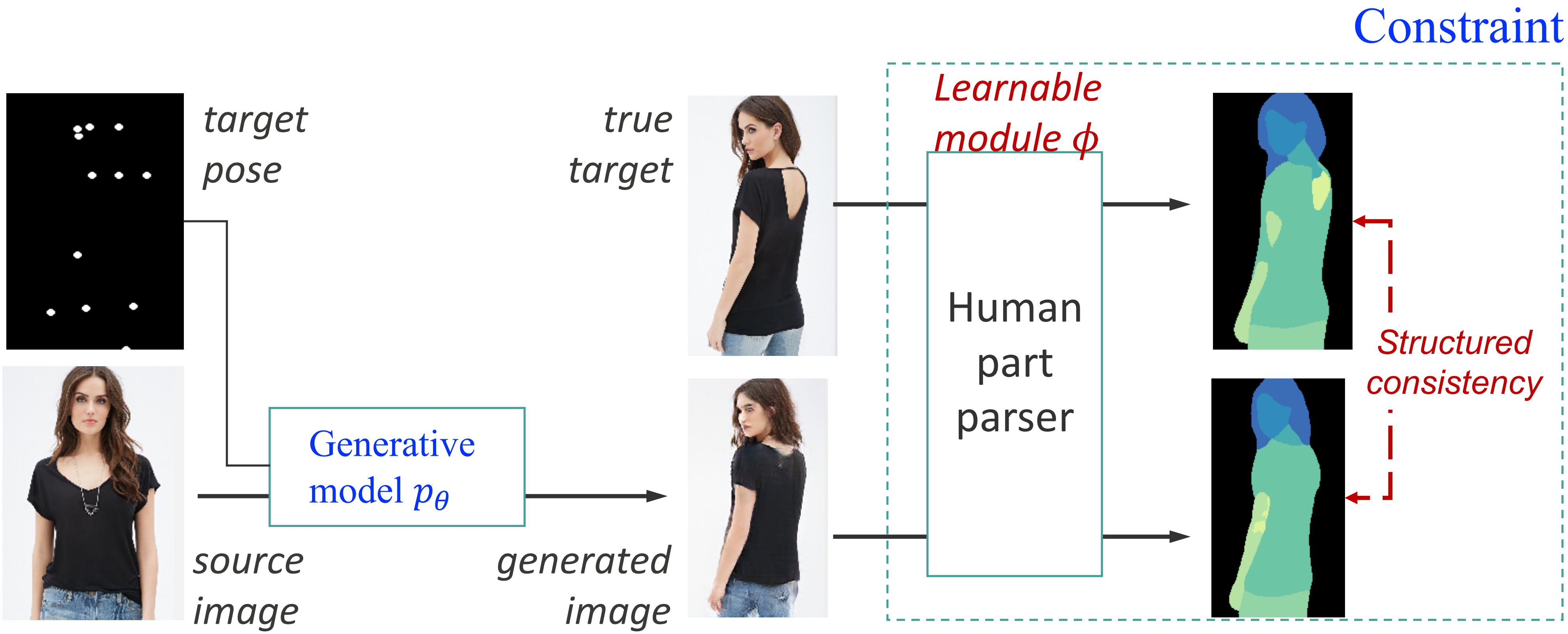
$$q^*(y|\boldsymbol{x}) = p_\theta(y|\boldsymbol{x}) \exp\left\{\sum_l \lambda_l r_l(y, \boldsymbol{x})\right\} / Z$$

- Learn the confidence value λ_l for each logical rule [Hu et al., 2016b]

$$q^*(\boldsymbol{x}) = p_\theta(\boldsymbol{x}) \exp\{\, \lambda f_\phi(\boldsymbol{x}) \,\}/Z$$

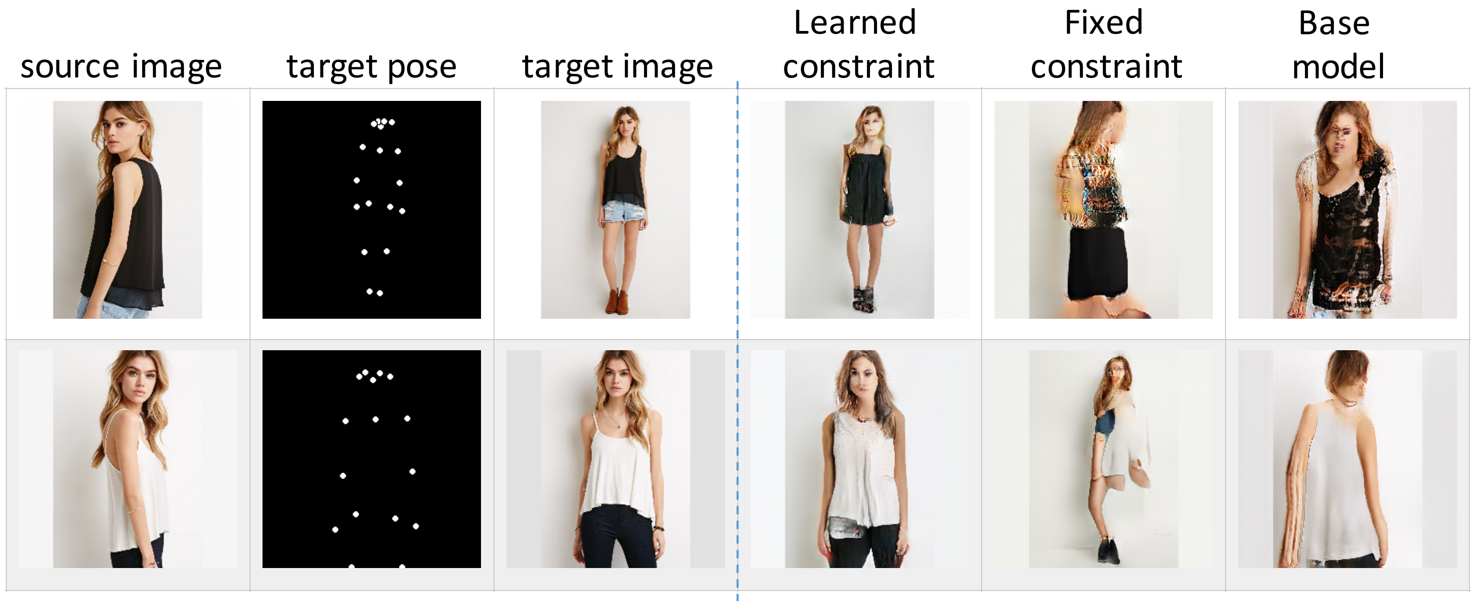
- More generally, optimize parameters of the constraint $f_\phi(\boldsymbol{x})$ [Hu et al., 2018]
  - Treat $f_\phi(\boldsymbol{x})$ as an extrinsic reward function
  - Use MaxEnt Inverse Reinforcement Learning to learn the "reward"

Pose-conditional Human Image Generation



[Hu et al., 2018]

Pose-conditional Human Image Generation



Samples generated by the models

	Method	SSIM	Human
1	Ma et al. [38]	0.614	—
2	Pumarola et al. [44]	0.747	—
3	Ma et al. [37]	0.762	—
4	Base model	0.676	0.03
5	With fixed constraint	0.679	0.12
6	With learned constraint	**0.727**	**0.77**

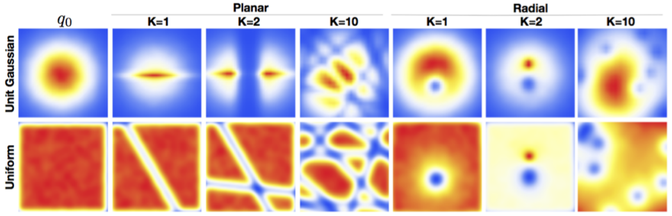
Quantitative and Human Evaluation

[Hu et al., 2018]

Takeaways

- Generative Adversarial Networks (GANs)
  - Wasserstein GAN: new learning objectives
  - Progressive GAN: new training schedule
  - BigGAN: scaling up GAN models

- Normalizing Flow (NF)
  - Chained transformation functions
  - Exact latent inference, density evaluation, sampling



- Integrating Domain Knowledge into Deep Learning
  - Domain knowledge as constraint
  - Learning rules / constraints