Adversarial Networks

Igor Gitman

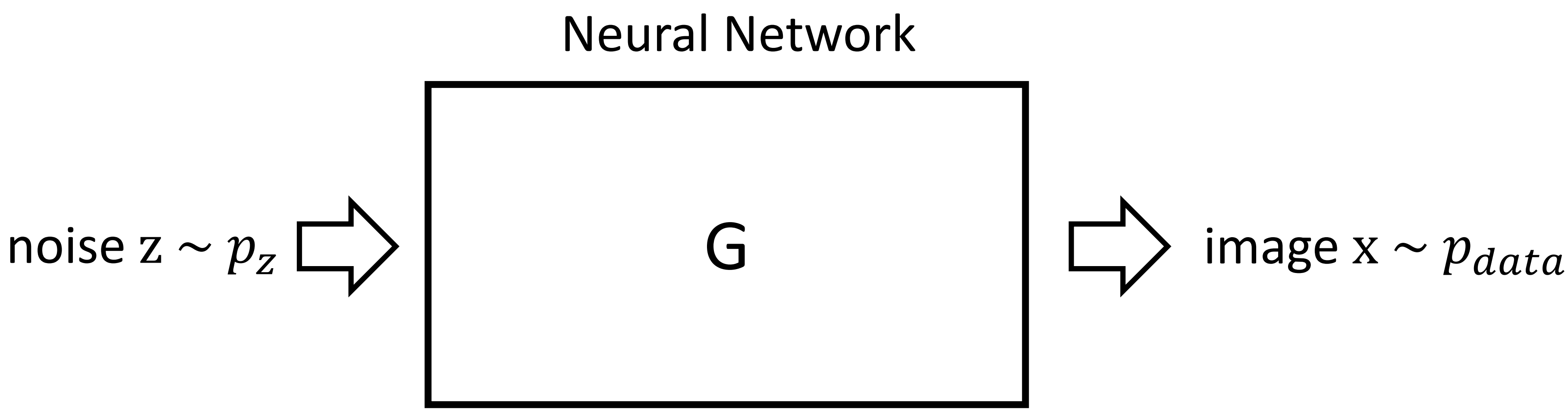
CMC MSU

Ian J. Goodfellow et al

Université de Montréal, Montréal

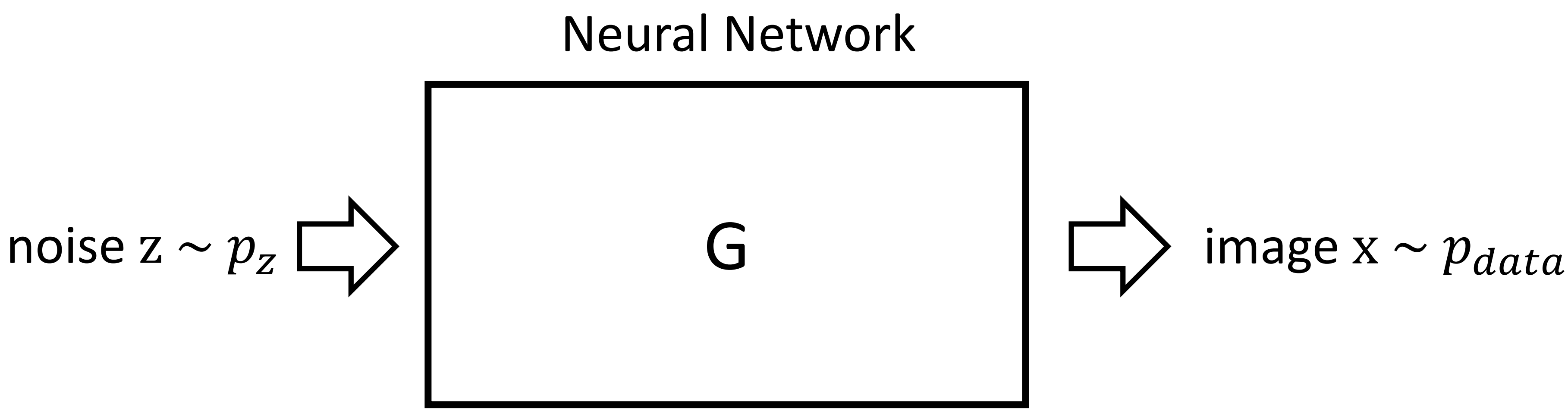
Generative adversarial nets

Objective: image generation



Generative adversarial nets

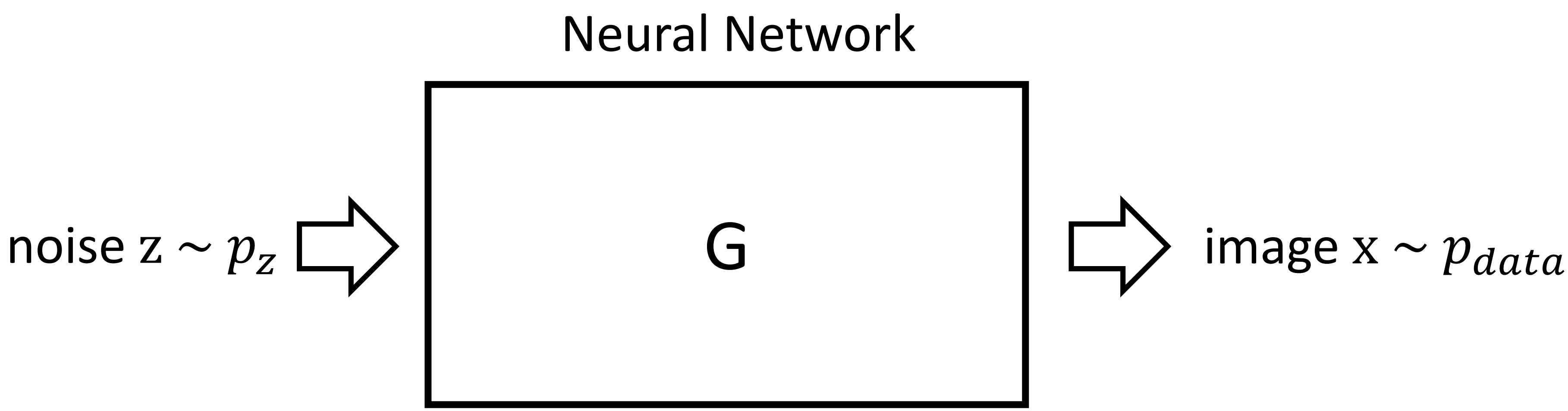
Objective: image generation



$$G \leftarrow \min_{\theta} \mathbb{E}_{z \sim p_z}\left[E\left(G(z;\theta)\right)\right]$$

Generative adversarial nets

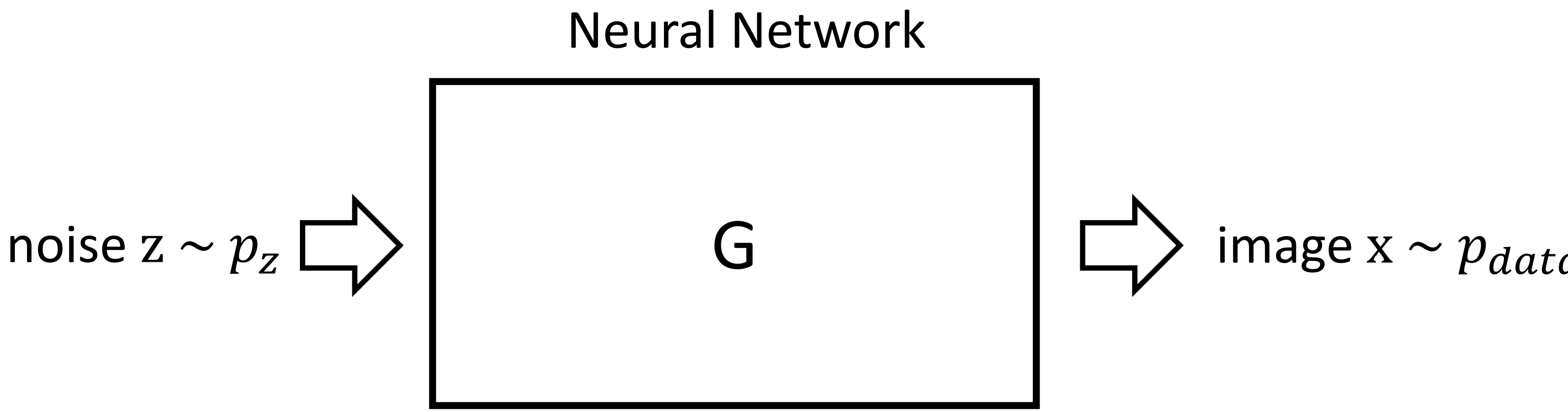
Objective: image generation



$$\mathrm{G} \longleftarrow \min_{\theta} \mathbb{E}_{z \sim p_z}\left[E\left(G(z;\theta)\right)\right] = \min_{\theta} \mathbb{E}_{z \sim p_z}[1 - P(G(z;\theta) \sim p_{data})]$$

Generative adversarial nets

Objective: image generation



$$\mathrm{G} \longleftarrow \min_{\theta} \mathbb{E}_{z \sim p_z}\Big[E\big(G(z;\theta)\big)\Big] = \min_{\theta} \mathbb{E}_{z \sim p_z}[1 - P(G(z;\theta) \sim p_{data})] =$$

$$= \min_{\theta} \mathbb{E}_{z \sim p_z}\Big[1 - D\big(G(z;\theta)\big)\Big]$$

D is another neural network!

Generative adversarial nets

$$\mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right] + \mathbb{E}_{x\sim p_{data}}[\log D(x)] \longrightarrow \min_G \max_D$$

Theoretical guarantees

$$\mathbb{E}_{z\sim p_z}\left[\log\left(1 - D\big(G(z)\big)\right)\right] + \mathbb{E}_{x\sim p_{data}}[\log D(x)] \longrightarrow \min_G \max_D$$

If we don't restrict G and D to parametric families, then:

1. For G fixed, the optimal discriminator D is

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

$p_g(x) = \int_{z:G(z)=x} p_z(z)dz$ – probability of x being an output of G

2. For $D = D_G^*$, minimum of the objective is achieved if and only if $p_g = p_{data}$

Theoretical guarantees

If we don't restrict G and D to parametric families, then:

If on each step of the iterative algorithm D is allowed to reach D_G^* and p_g is updated so as to decrease

$$\mathbb{E}_{x \sim p_g}\left[\log\left(1 - D(x)\right)\right] + \mathbb{E}_{x \sim p_{data}}[\log D(x)]$$

then p_g converges to p_{data}

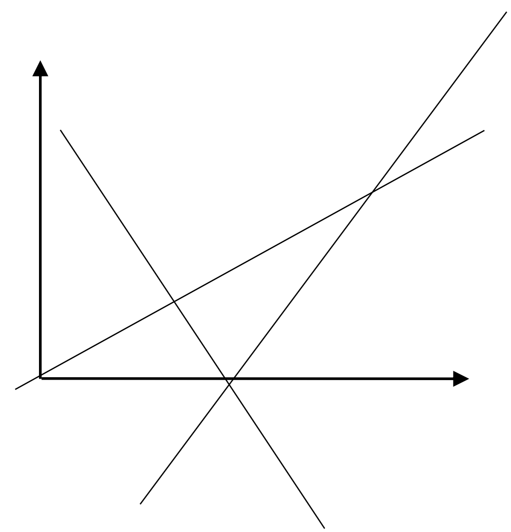
Theoretical guarantees

$$V(p_g, D) = \mathbb{E}_{x \sim p_g}[\log(1 - D(x))] + \mathbb{E}_{x \sim p_{data}}[\log D(x)]$$

$V(p_g, D)$ is convex (linear) in p_g $\forall$ D

$$\mathrm{C}(p_g) = \max_D \left[\mathbb{E}_{x \sim p_g}[\log(1 - D(x))] + \mathbb{E}_{x \sim p_{data}}[\log D(x)]\right] \Rightarrow$$

$$\Rightarrow \ \partial V(p_g, D^*) \in \partial C(p_g)$$



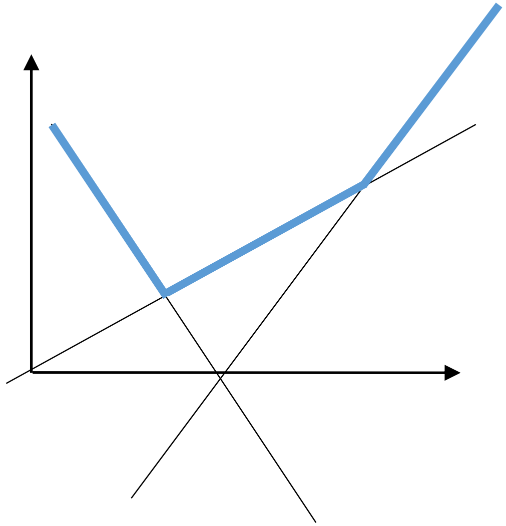
Theoretical guarantees

$$V(p_g, D) = \mathbb{E}_{x\sim p_g}[\log(1 - D(x))] + \mathbb{E}_{x\sim p_{data}}[\log D(x)]$$

$V(p_g, D)$ is convex (linear) in p_g $\forall$ D

$$\mathrm{C}(p_g) = \max_D \left[\mathbb{E}_{x\sim p_g}[\log(1 - D(x))] + \mathbb{E}_{x\sim p_{data}}[\log D(x)]\right] \Rightarrow$$

$$\Rightarrow \ \partial V(p_g, D^*) \in \partial C(p_g)$$



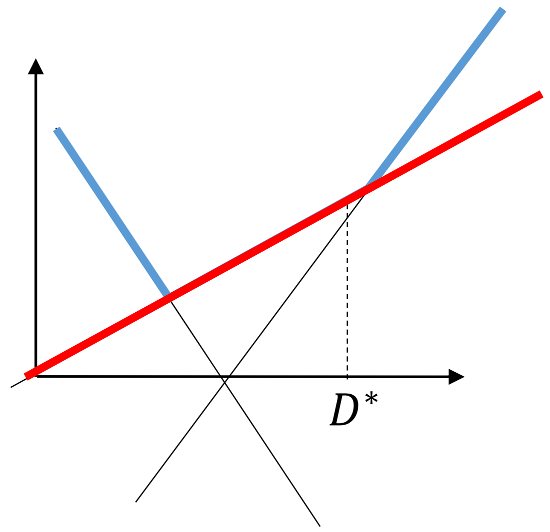
Theoretical guarantees

$$V(p_g, D) = \mathbb{E}_{x \sim p_g}[\log(1 - D(x))] + \mathbb{E}_{x \sim p_{data}}[\log D(x)]$$

$V(p_g, D)$ is convex (linear) in p_g $\forall\, D$

$$\mathrm{C}(p_g) = \mathbb{E}_{x \sim p_g}[\log(1 - D^*(x))] + \mathbb{E}_{x \sim p_{data}}[\log D^*(x)] \Rightarrow$$

$$\Rightarrow\ \partial V(p_g, D^*) \in \partial C(p_g)$$



Training algorithm

for T iterations **do**
  for k steps **do**
    sample $\{z^{(1)}, \dots, z^{(m)}\} \sim p_z$
    sample $\{x^{(1)}, \dots, x^{(m)}\} \sim p_{data}$
    update $D(x; \theta_d)$:

$$v \coloneqq \mu v + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}; \theta_d) + \log\left(1 - D\left(G\left(z^{(i)}\right); \theta_d\right)\right) \right]$$

$$\theta_d \coloneqq \theta_d + v$$

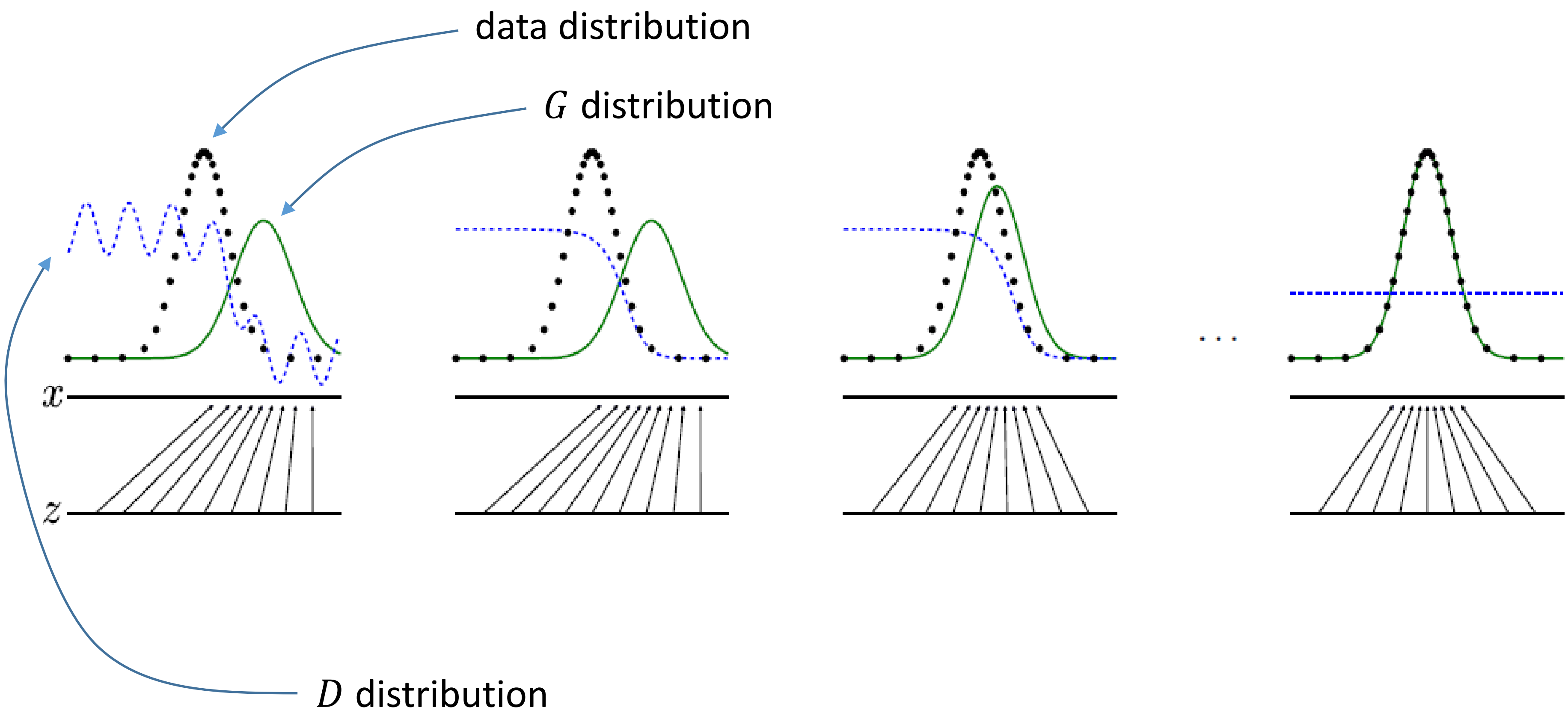
  end for
  sample $\{z^{(1)}, \dots, z^{(m)}\} \sim p_z$
  update G$(x; \theta_g)$:

$$v \coloneqq \mu v - \alpha \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}; \theta_g\right)\right)\right) \right]$$

$$\theta_g \coloneqq \theta_g + v$$

end for

Generative adversarial nets

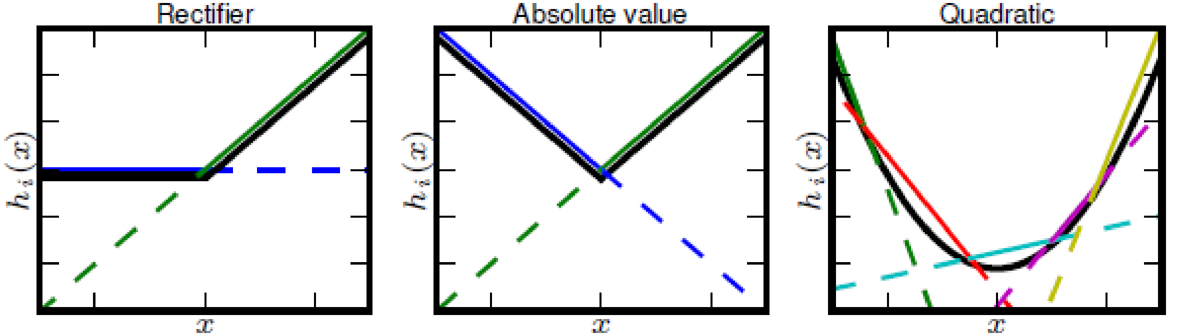


Important details

1. Maxout activation functions.

$$h_i(x) = \max_{j \in [1,k]} \left[(x^T W)_{ij} + b_{ij} \right]$$

$$h \in \mathbb{R}^m, x \in \mathbb{R}^d, W \in \mathbb{R}^{d \times m \times k}, \mathrm{b} \in \mathbb{R}^{m \times k}$$



Important details

1. Maxout activation functions.
2. Early in training, $\log[1 - D(G(z))]$ saturates => train G to maximize $\log[D(G(z))]$.
3. Desynchronization of D and G (don't train G too much without updating D).

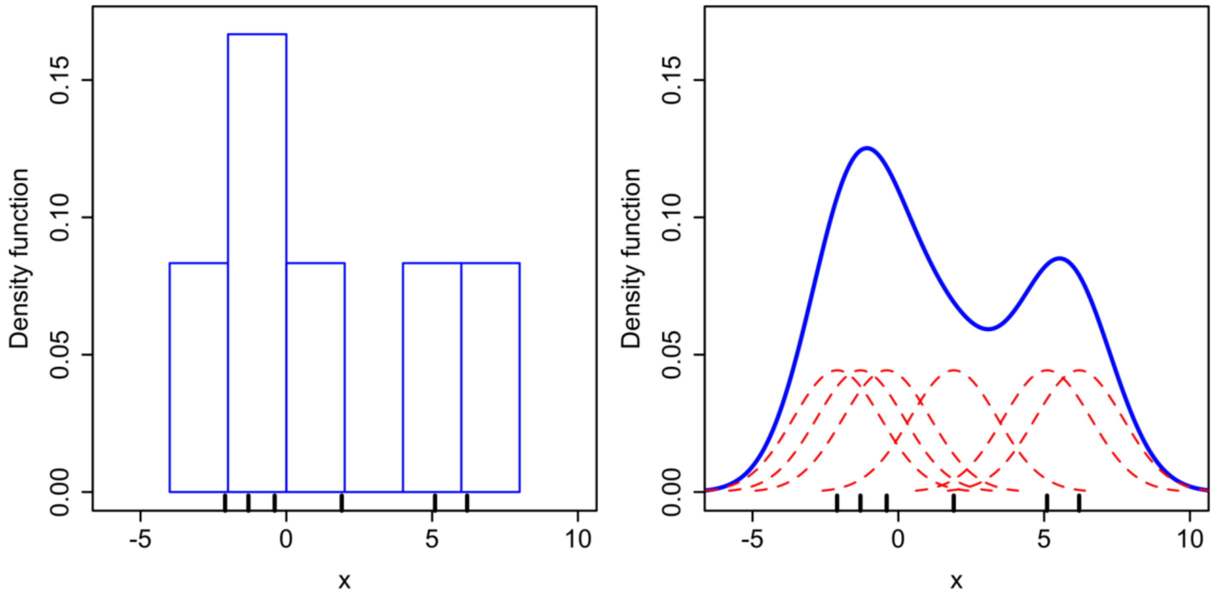
Experiments

1. Quality assessment using Gaussian Parzen windows.

Experiments

1. Quality assessment using Gaussian Parzen windows.

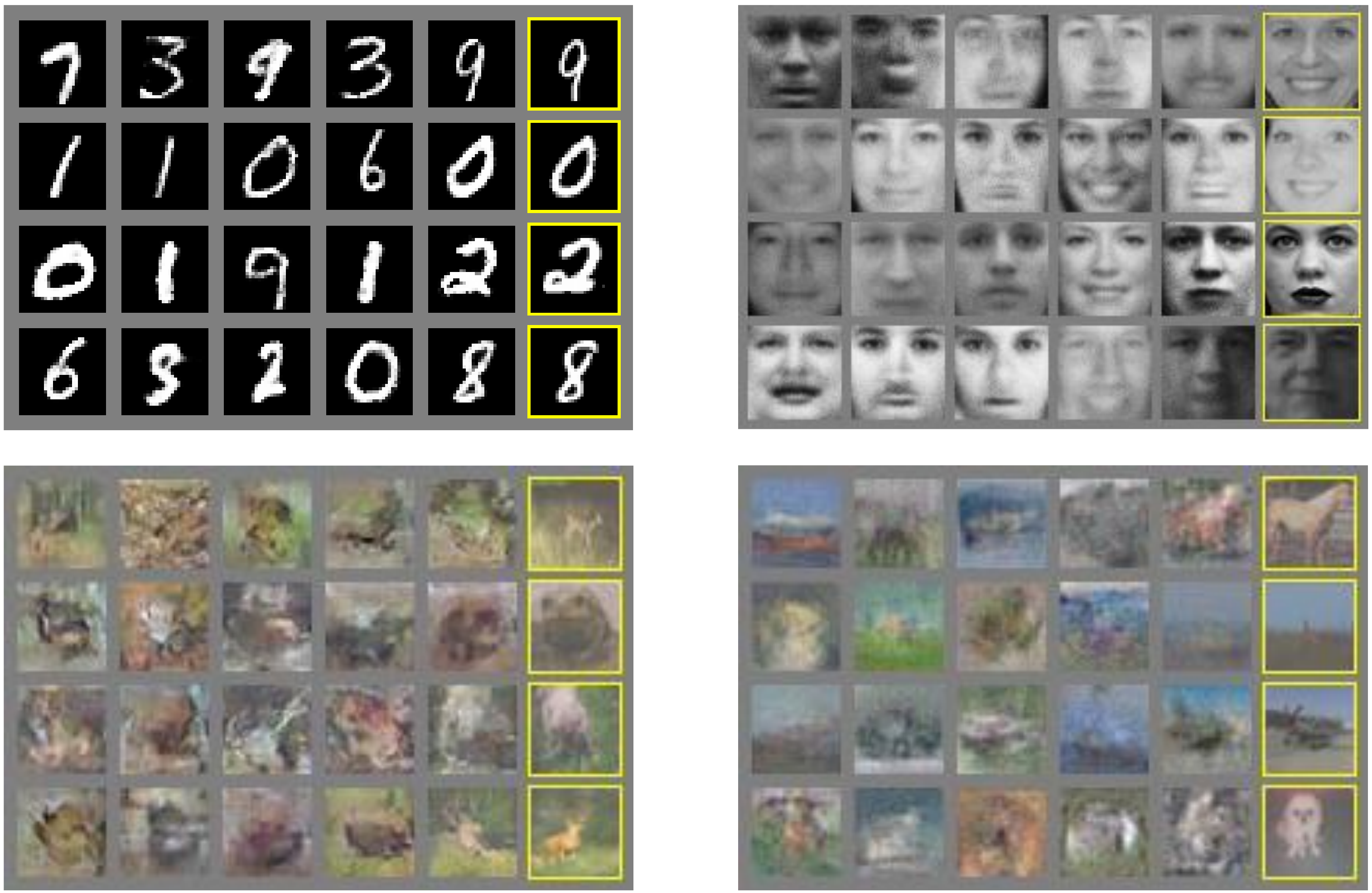
$$f_h(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\sqrt{2\pi}h} e^{\frac{-(x-x_i)^2}{2h^2}}$$



Experiments

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Experiments



Laplacian Pyramid of Adversarial Networks

Conditional adversarial networks:

$$\mathbb{E}_{z\sim p_z, l\sim p_l}\left[\log\left(1 - D(G(z,l),l)\right)\right] + \mathbb{E}_{x,l\sim p_{data}}[\log D(x,l)] \longrightarrow \min_G \max_D$$

$d(I)$ – downsampling operator, $u(I)$ – upsampling operator

$[I_0, I_1, \ldots, I_K]$ – consequentially applying $d(I)$

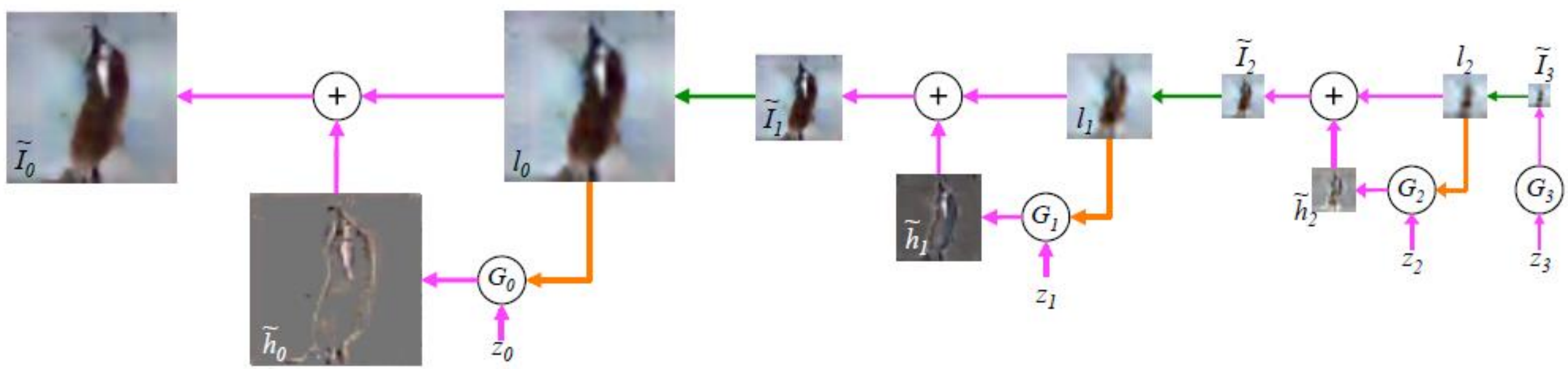
$$h_k = I_k - u(I_{k+1}) \Rightarrow I_k = u(I_{k+1}) + h_k$$

Let's learn h_k using adversarial pair G_k, D_k, conditioned on $u(I_{k+1})$

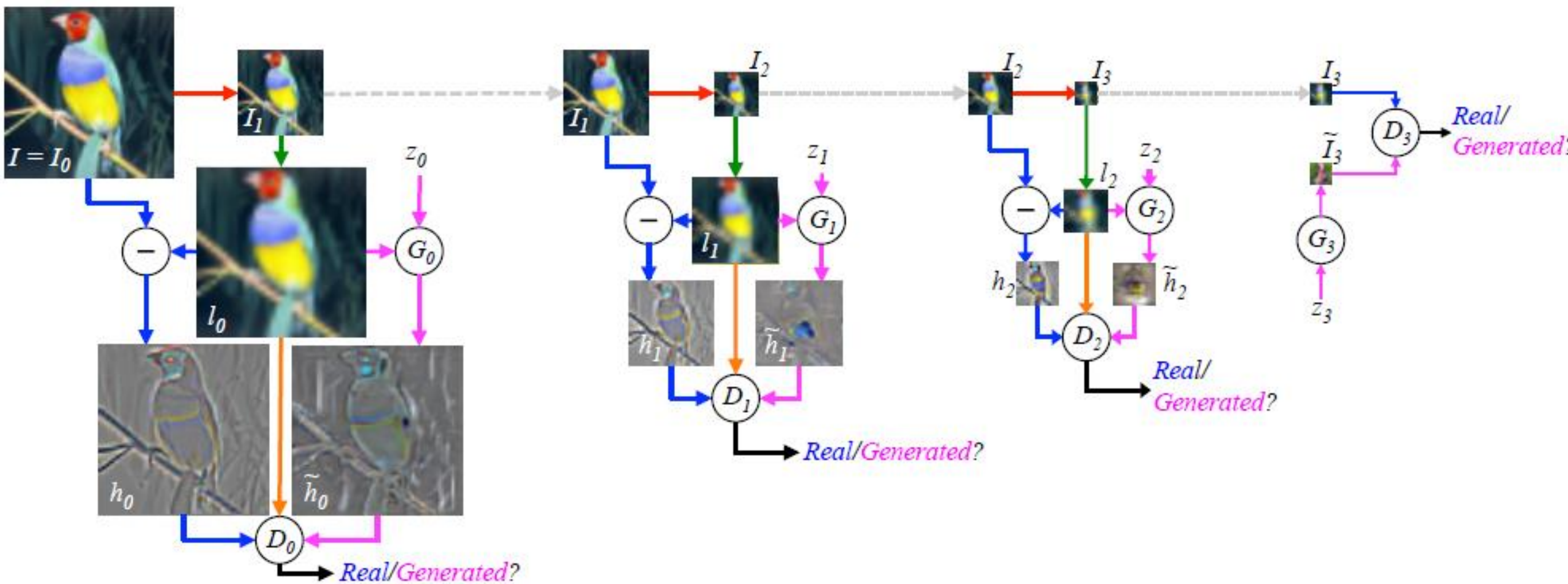
Laplacian Pyramid of Adversarial Networks

Conditional adversarial networks:

$$\mathbb{E}_{z\sim p_z, l\sim p_l}\left[\log\left(1 - D(G(z,l),l)\right)\right] + \mathbb{E}_{x,l\sim p_{data}}[\log D(x,l)] \longrightarrow \min_G \max_D$$

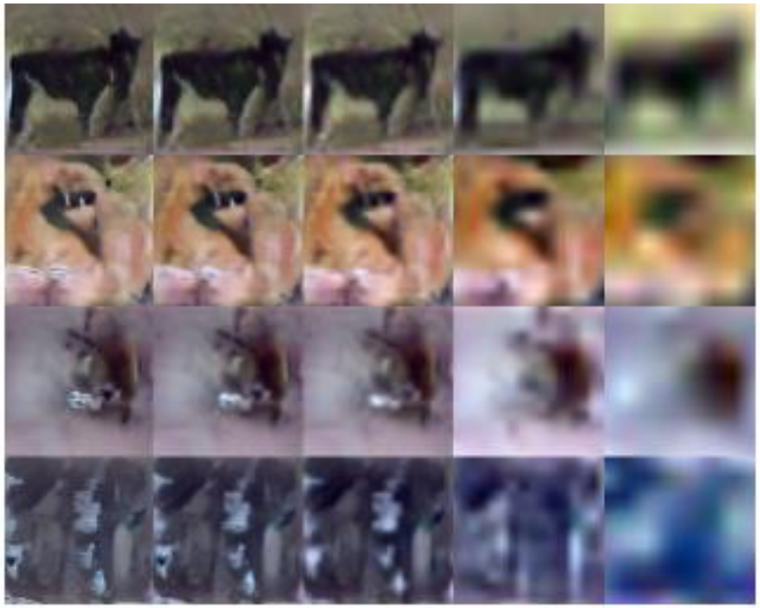


Laplacian Pyramid of Adversarial Networks



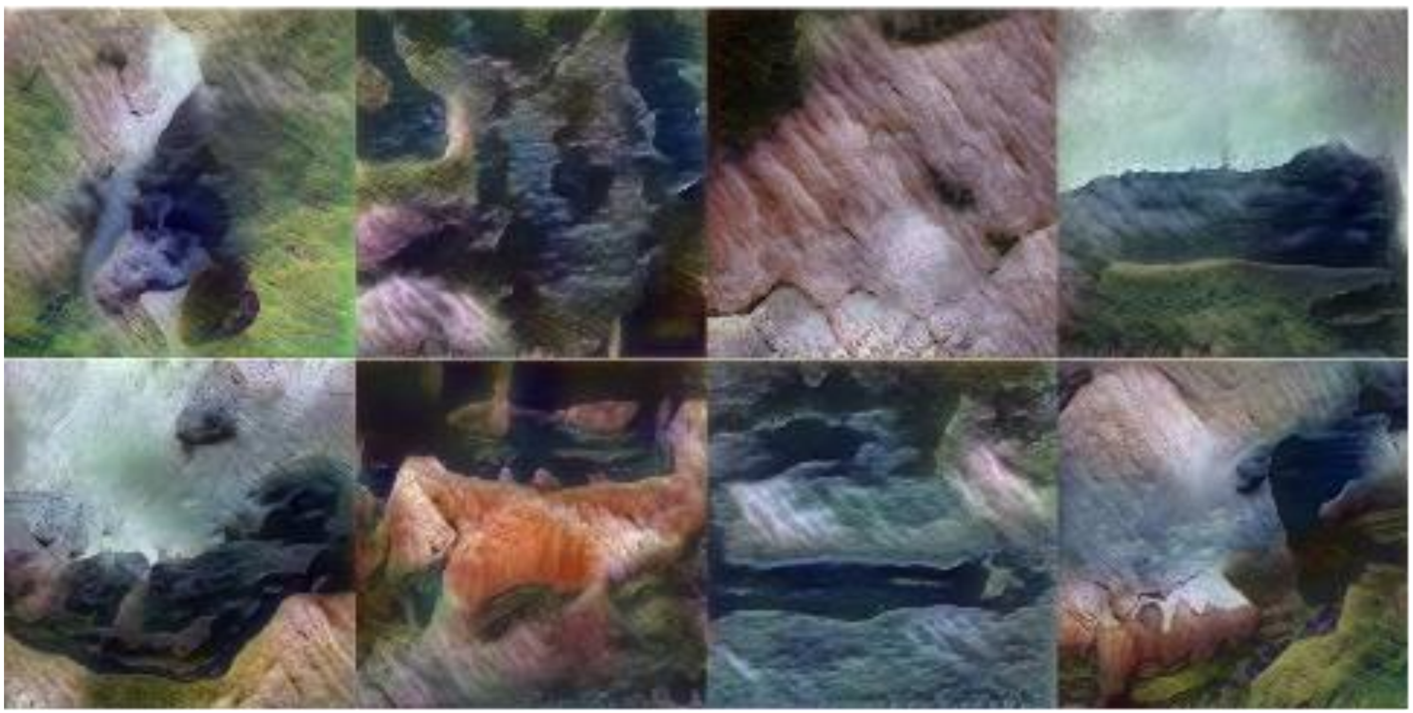
Giving up any "global" notion of fidelity!

Laplacian Pyramid of Adversarial Networks



Laplacian Pyramid of Adversarial Networks

Laplacian Pyramid of Adversarial Networks



soumith.ch/eyescream

Image deconvolution using Adversarial Networks

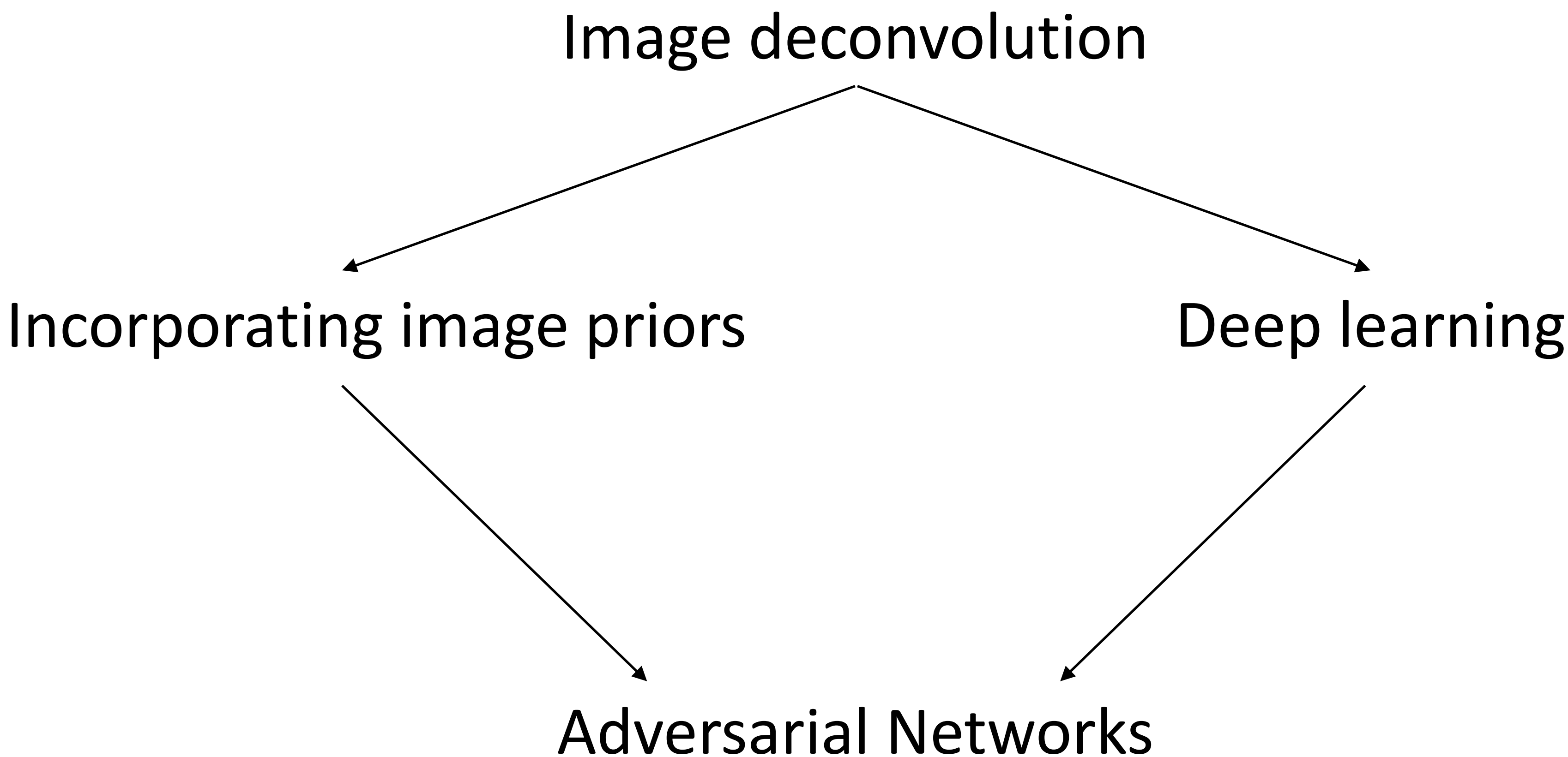


Image deconvolution using Adversarial Networks