

The Adversarial Networks Nonsense

Soumith Chintala
Facebook AI Research

Recap of Lecture 16/17

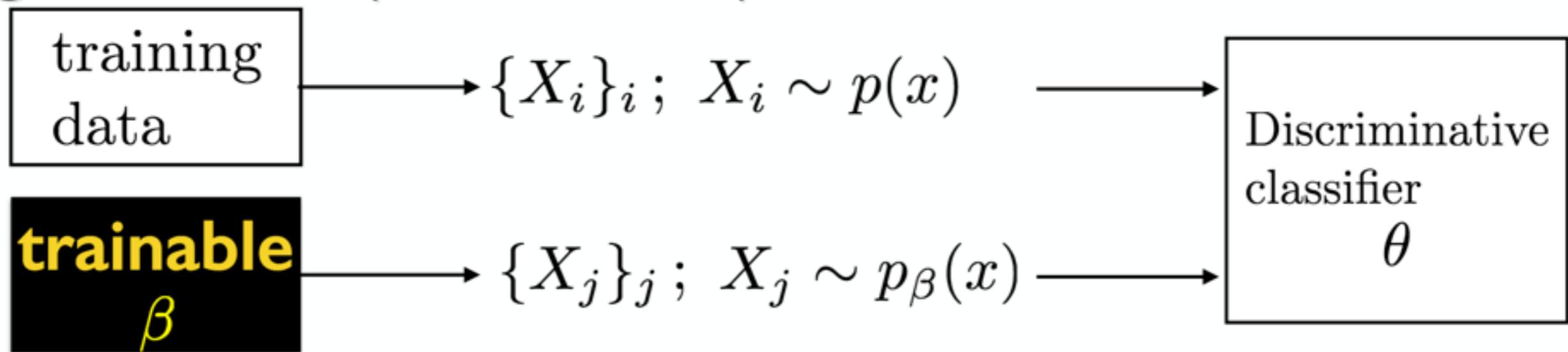
Generative Adversarial Networks

[Goodfellow et al., '14]

- Suppose we have a *trainable* black box generator:



- Given observed data $\{X_i\}_i$; $X_i \sim p(x)$, how to force our generator to produce samples from $p(x)$?

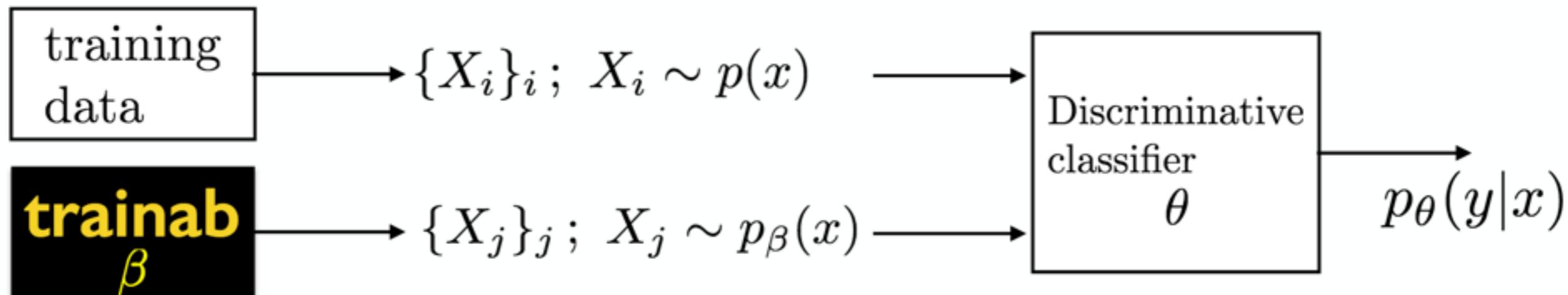


- The generator should make the classification task as *hard* as possible for any discriminator.

Generative Adversarial Networks

[Goodfellow et al., '14]

- Train generator and discriminator in a minimax setting:



$y = 1$: “real” samples

$y = 0$: “fake” samples

$$\min_{\beta} \max_{\theta} \left(\mathbb{E}_{x \sim p_{data}} \log p_{\theta}(y=1|x) + \mathbb{E}_{x \sim p_{\beta}} \log p_{\theta}(y=0|x) \right).$$

Generative Adversarial Networks

- Q: Do we have consistency? (in the limit of infinite capacity)

Given current p_β and p_{data} , the optimum discriminator is given by

$$D(x) = p(y = 1|x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} .$$

For each x ,

$$p_{\text{data}}(x) \log D(x) + p_\beta(x) \log(1 - D(x)) = (p_{\text{data}}(x) + p_\beta(x)) (\alpha \log \gamma + (1 - \alpha) \log(1 - \gamma)) ,$$

$$\alpha = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\beta(x)} , \quad \gamma = D(x) .$$

But

$$\alpha \log \gamma + (1 - \alpha) \log(1 - \gamma) = -H(\bar{\alpha}) - D_{KL}(\bar{\alpha} || p(y|x)) \leq -H(\bar{\alpha})$$

Generative Adversarial Networks

- It results that

$\min -H(\bar{\alpha})$ is attained when $\alpha = 1/2$, thus

$$p_\beta(x) = p_{data}(x)$$

- In practice, however, we parametrize both generator and discriminator using neural networks.
- Optimize the cost using gradient descent.

Generative Adversarial Training

$$F(\beta, \theta) = (\mathbb{E}_{x \sim p_{data}} \log p_\theta(y=1|x) + \mathbb{E}_{x \sim p_\beta} \log p_\theta(y=0|x)) .$$

$$\min_{\beta} \max_{\theta} F(\beta, \theta)$$

- Challenge: it is unfeasible to optimize fully in the inner discriminator loop:

$$\theta^*(\beta) = \arg \max_{\theta} F(\beta, \theta) . \quad G(\beta) := F(\beta, \theta^*(\beta))$$

- Indeed,
- Numerical approach: alternate k steps of discriminator update with 1 step of generator update.

LAPGAN

[Denton, Chintala et al.'15]

- Initial GAN models were hard to scale to large input domains.
- Laplacian Pyramid of Adversarial Networks significantly improved quality by generating independently at each scale.
- Laplacian Pyramids are invertible linear multi-scale decompositions:

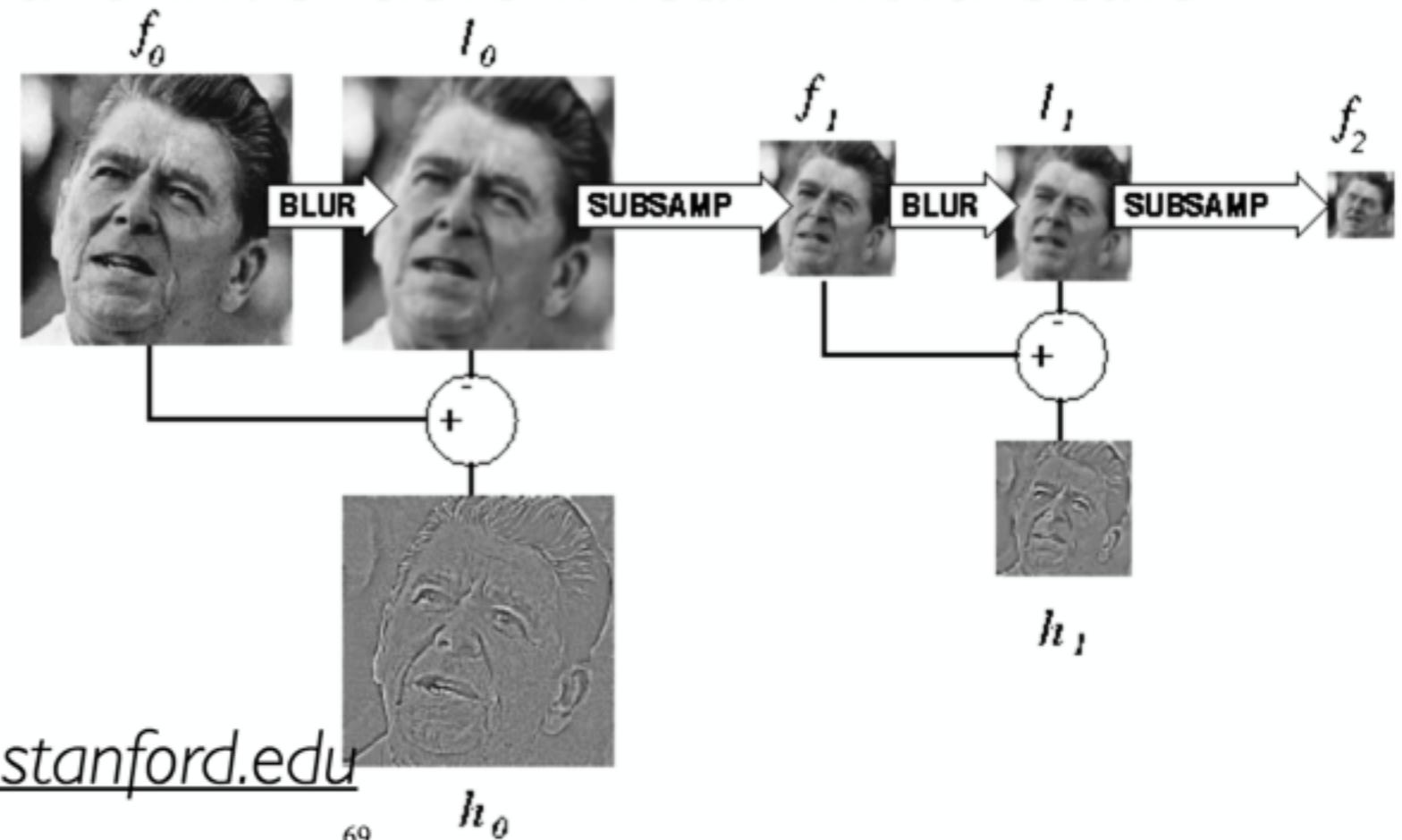
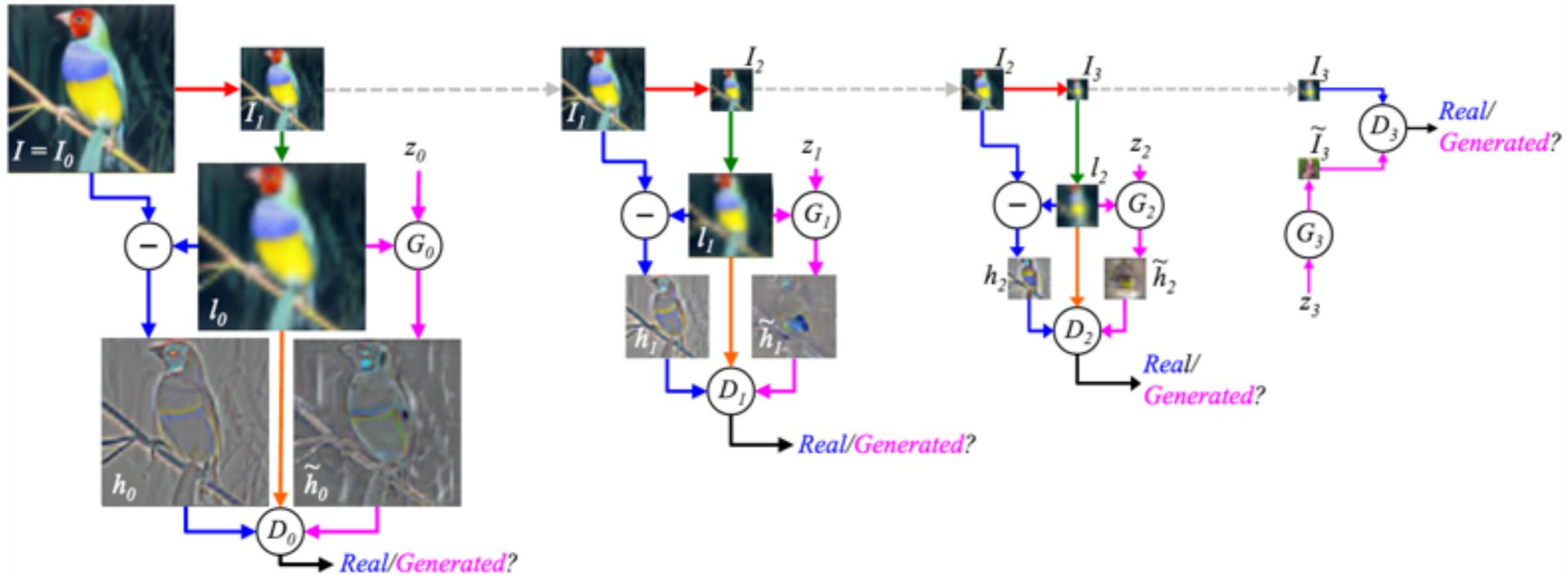


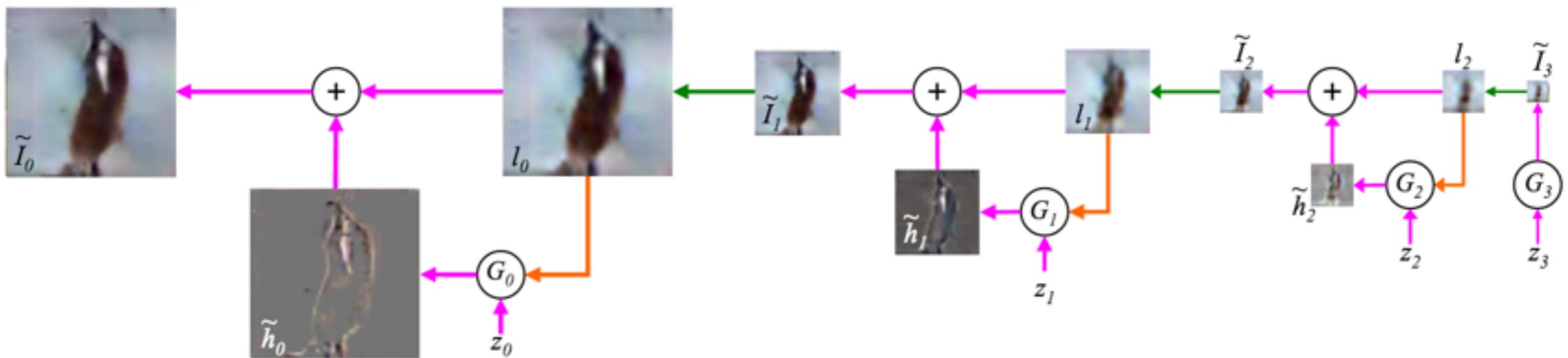
figure source: <http://sepwww.stanford.edu>

LAPGAN

- Training procedure:



- Sampling procedure:



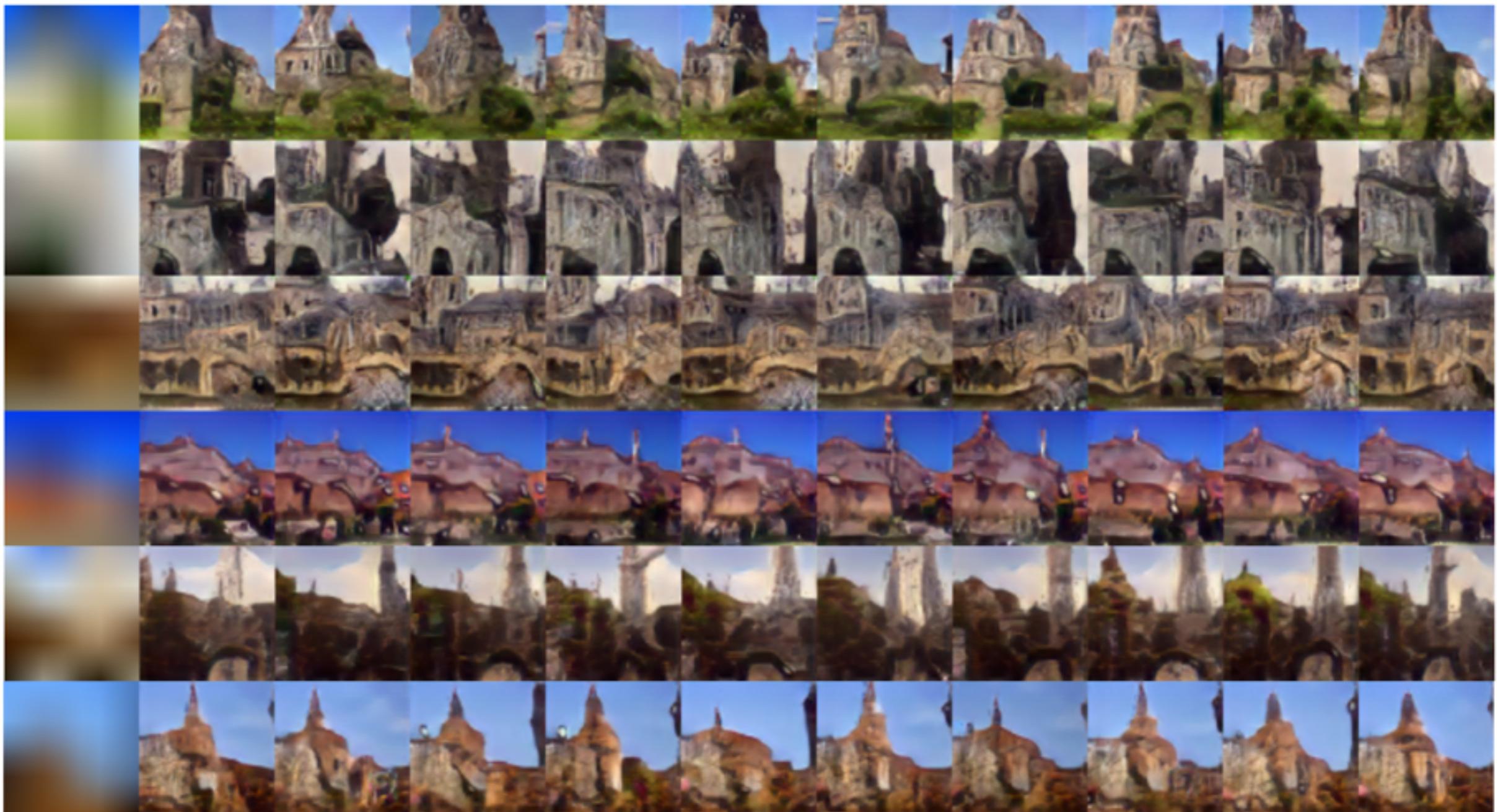
LAPGAN

- Samples generated from the model:



LAPGAN

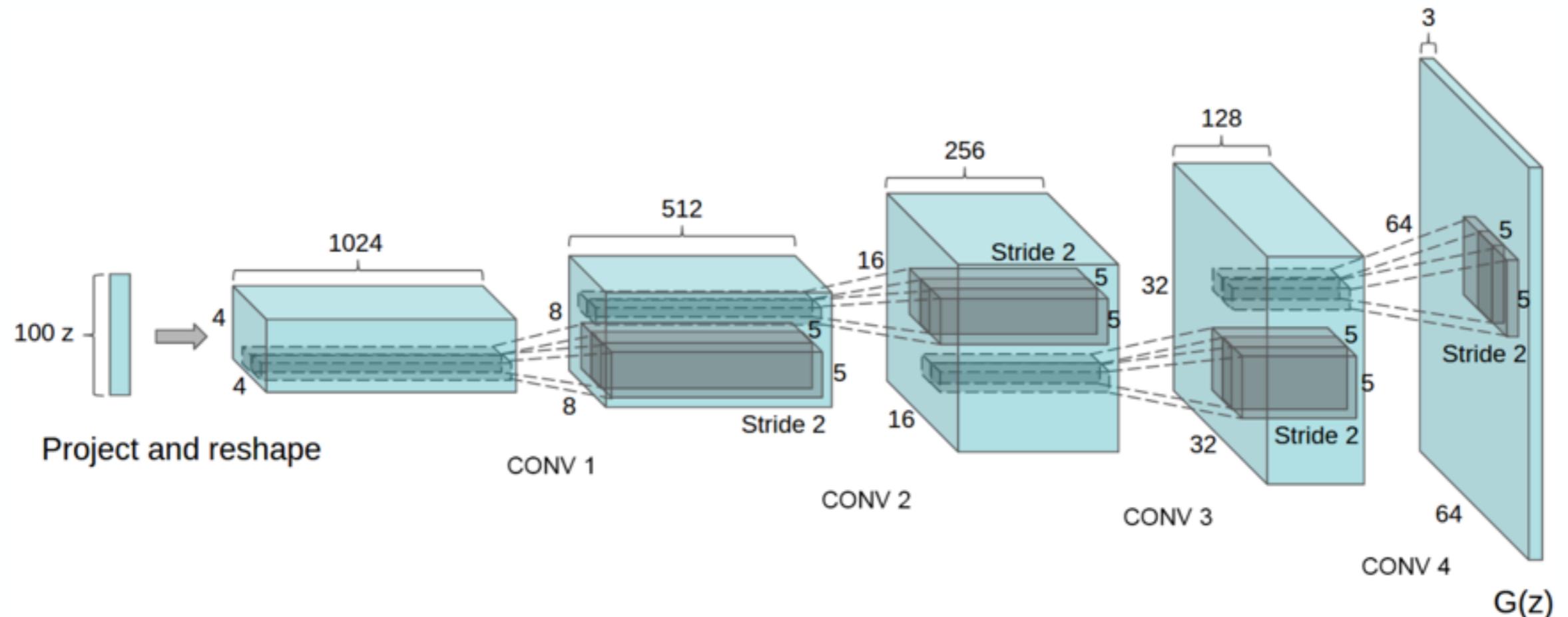
- Samples generated from the model:



DC-GAN

[Radford et al.'16]

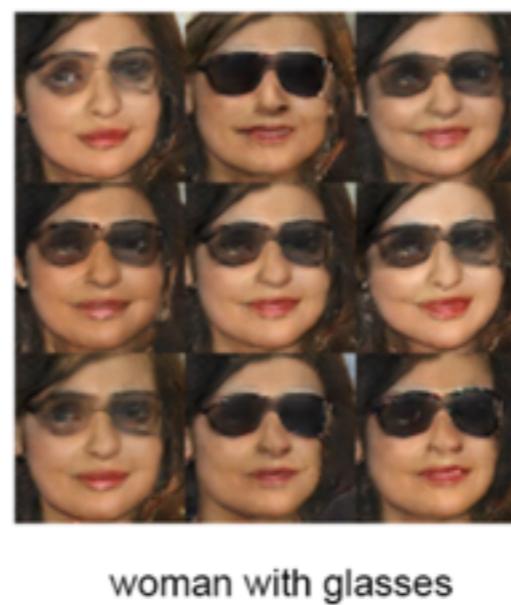
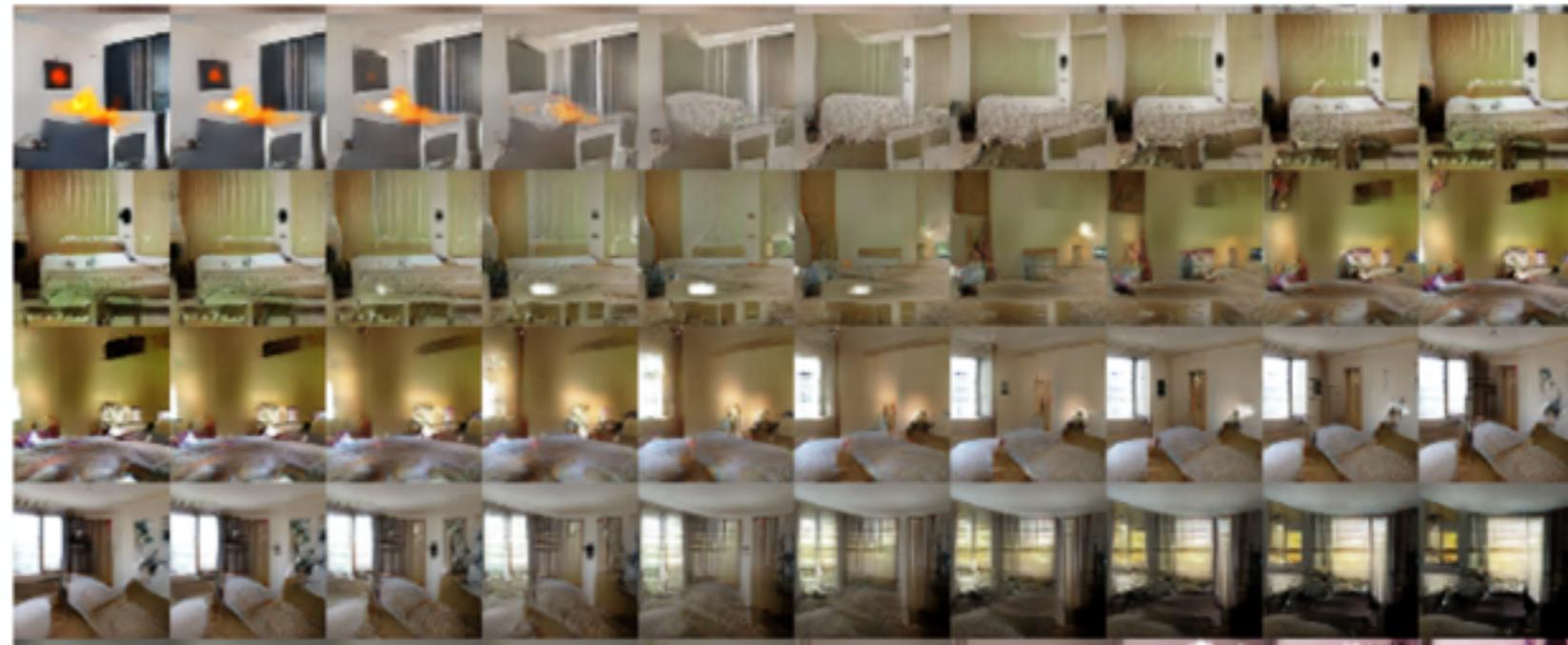
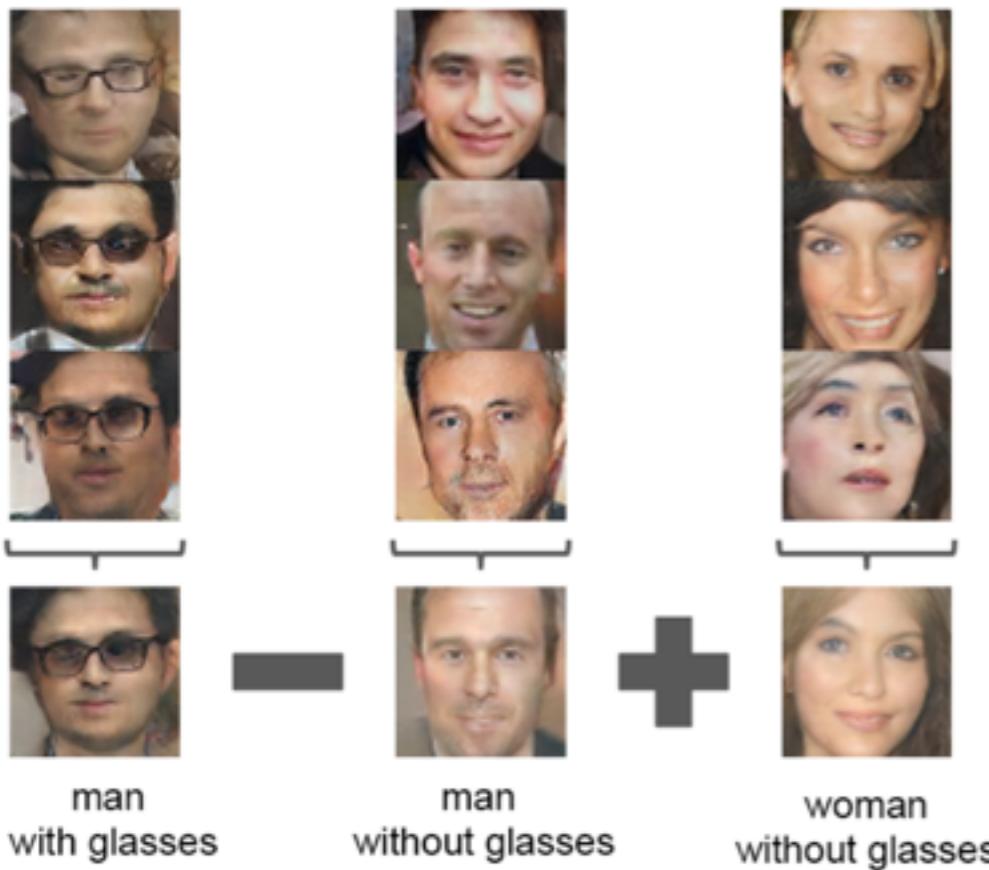
- Improved multi-scale architecture and Batch-Normalization:



DC-GAN

- Improved multi-scale architecture and Batch-Normalization:

[Radford et al.'16]



Generative Adversarial Networks

- GRAN [Generative Recurrent Adversarial Nets, Im et al.'16]
- Video Prediction [Mathieu et al.'16]
- CNN Reconstruction [Brox et al.'16]
- A very *hot* topic within the Deep Learning community



Generative Adversarial Networks

- Some open research directions:

1. **Optimization:**

1. How to ensure a correct algorithm?
2. Existence of a Lyapunov function?

2. **Statistics:**

1. How to determine the discriminator power (eg VC-dimension) to obtain consistent estimators?
2. Control of overfitting to the training distribution?

3. **Applications:**

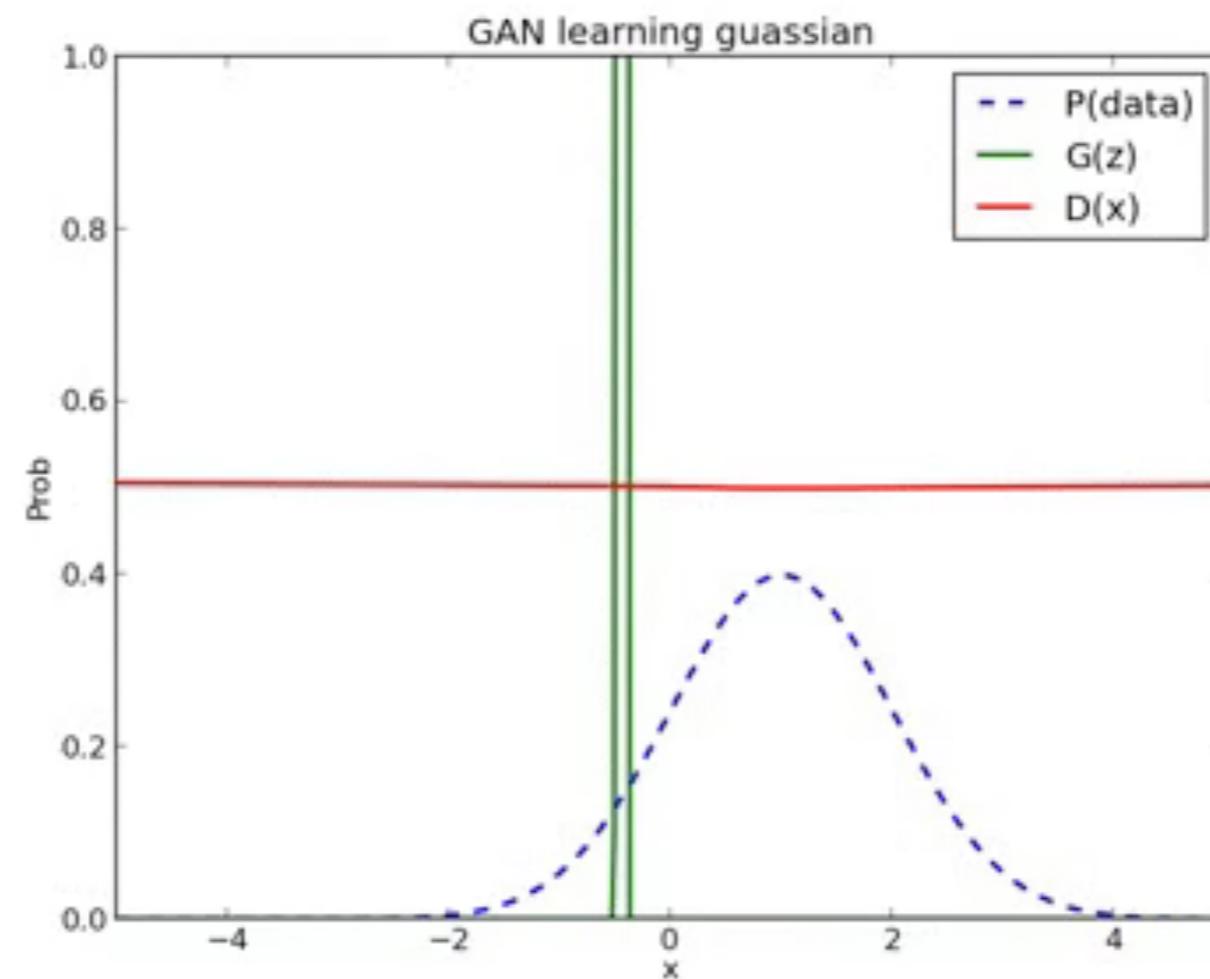
- Language Modeling
- Reinforcement Learning
- Algorithmic Tasks
- Importance Sampling

GANs and their Properties

What do GANs Optimize?

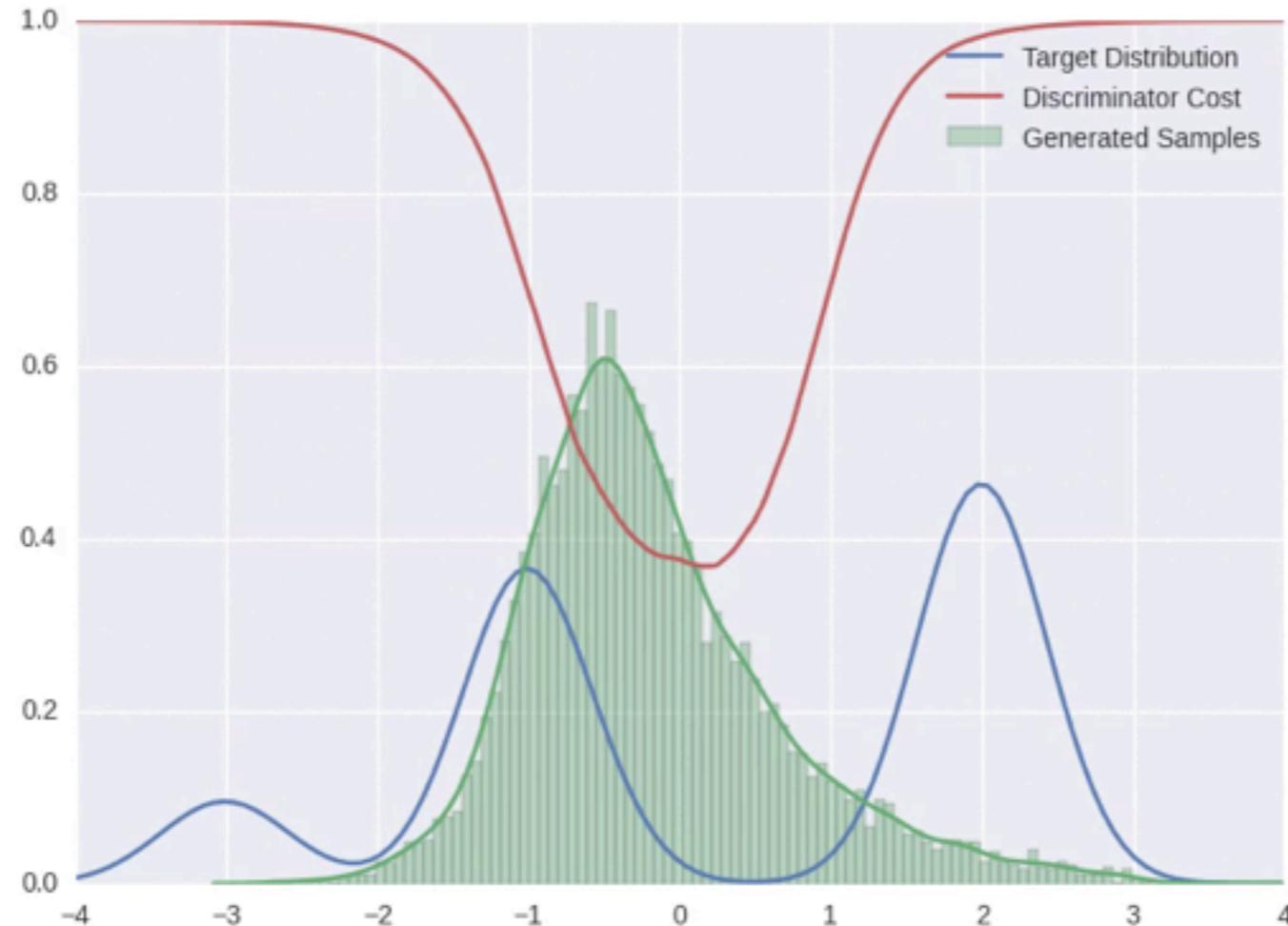
Open problem: structured latent distributions

Some visualizations



<https://twitter.com/AlecRad/status/619605092522676225>

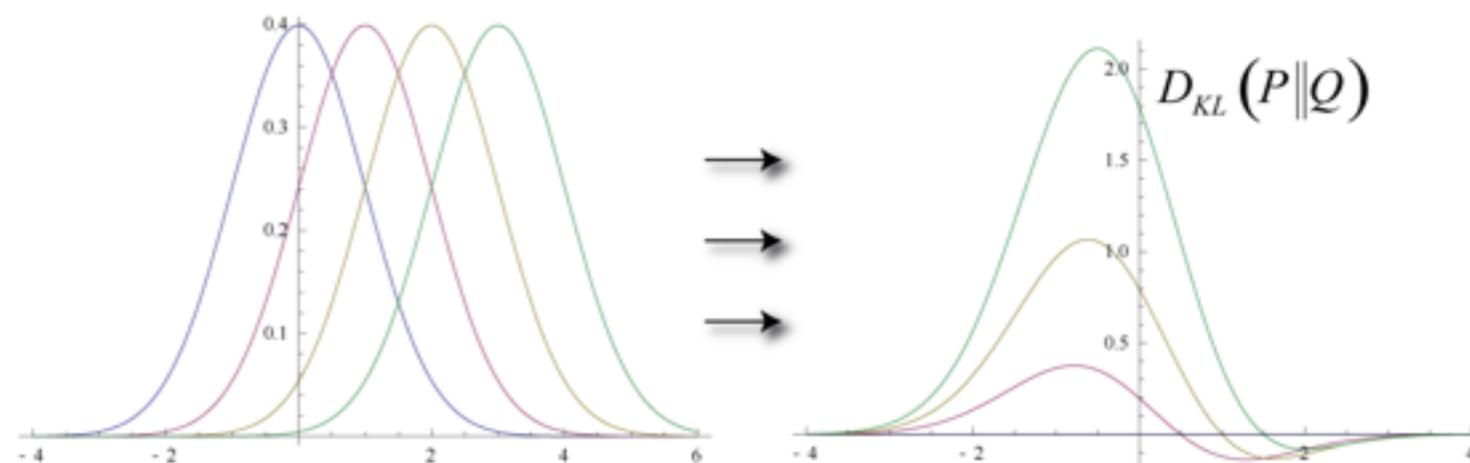
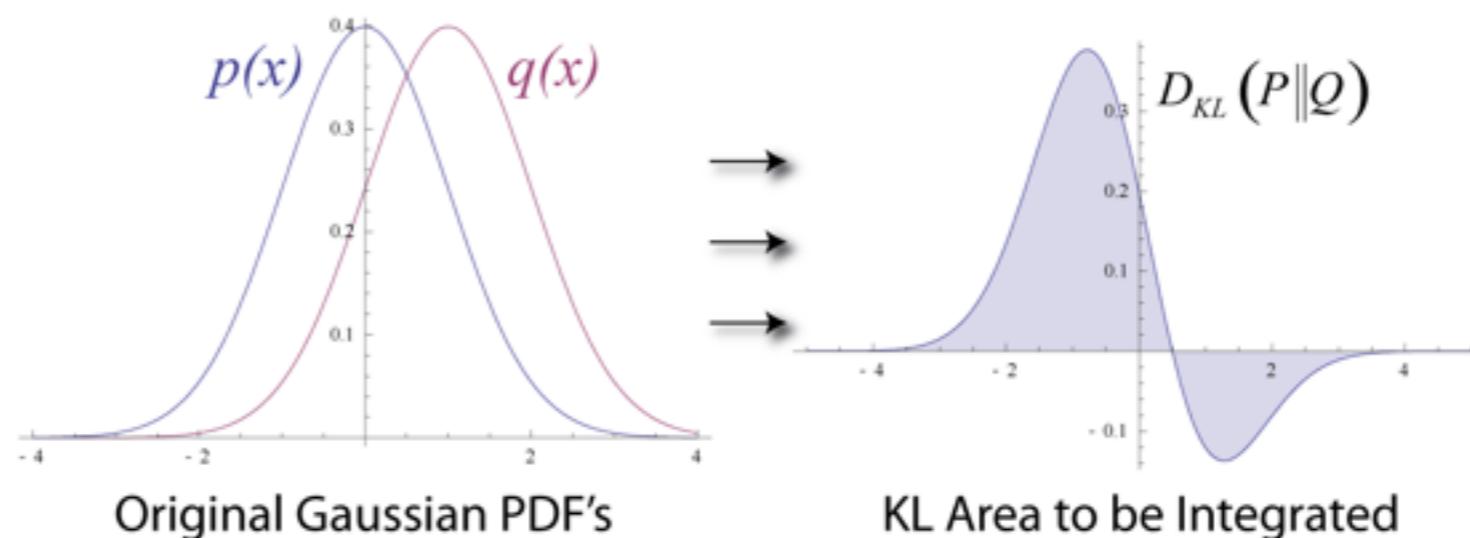
Some visualizations



https://twitter.com/Luke_Metz/status/711744051918282752

KL-Divergence

$$D_{\text{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx,$$



KL-Divergence: Issues

Whiteboard session

KL-Divergence: Issues

- Autoencoders optimizing $KL(P/Q)$
 - Blurry



JS-Divergence

$$\text{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

$$M = \frac{1}{2}(P + Q)$$

$$D(P \parallel Q) = \text{KL-Divergence} = D_{\text{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx,$$

JS-Divergence

Whiteboard session

JS-Divergence

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{\text{data}}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D_G^*(x) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(x) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{x \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{x \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$C(G) = -\log(4) + KL \left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2} \right) + KL \left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2} \right) \quad (5)$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model’s distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD(p_{\text{data}} \| p_g) \quad (6)$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process. \square

“Generative Adversarial Nets” - Goodfellow et. al.

λ -Divergence

Symmetrised divergence [edit]

Kullback and Leibler themselves actually defined the divergence as:

$$D_{\text{KL}}(P\|Q) + D_{\text{KL}}(Q\|P)$$

which is symmetric and nonnegative. This quantity has sometimes been used for feature selection in [classification](#) problems, where P and Q are the conditional [pdfs](#) of a feature under two different classes.

An alternative is given via the λ divergence,

$$D_\lambda(P\|Q) = \lambda D_{\text{KL}}(P\|\lambda P + (1 - \lambda)Q) + (1 - \lambda)D_{\text{KL}}(Q\|\lambda P + (1 - \lambda)Q),$$

which can be interpreted as the expected information gain about X from discovering which probability distribution X is drawn from, P or Q , if they currently have probabilities λ and $(1 - \lambda)$ respectively.

The value $\lambda = 0.5$ gives the [Jensen–Shannon divergence](#), defined by

$$D_{\text{JS}} = \frac{1}{2}D_{\text{KL}}(P\|M) + \frac{1}{2}D_{\text{KL}}(Q\|M)$$

where M is the average of the two distributions,

$$M = \frac{1}{2}(P + Q).$$

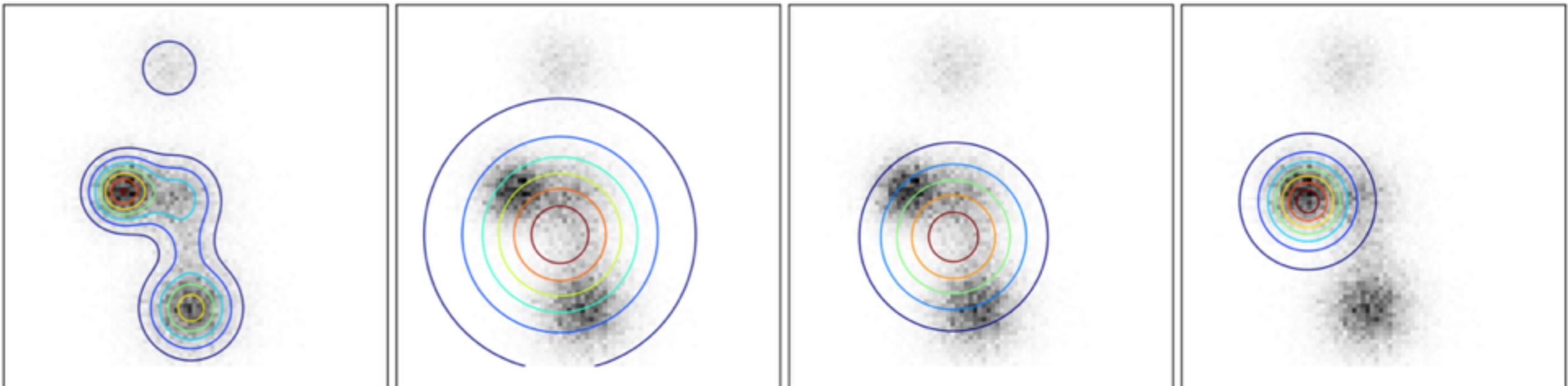
λ -Divergence

A: P

B: $\arg \min_Q JS_{0.1}[P\|Q]$

C: $\arg \min_Q JS_{0.5}[P\|Q]$

D: $\arg \min_Q JS_{0.99}[P\|Q]$



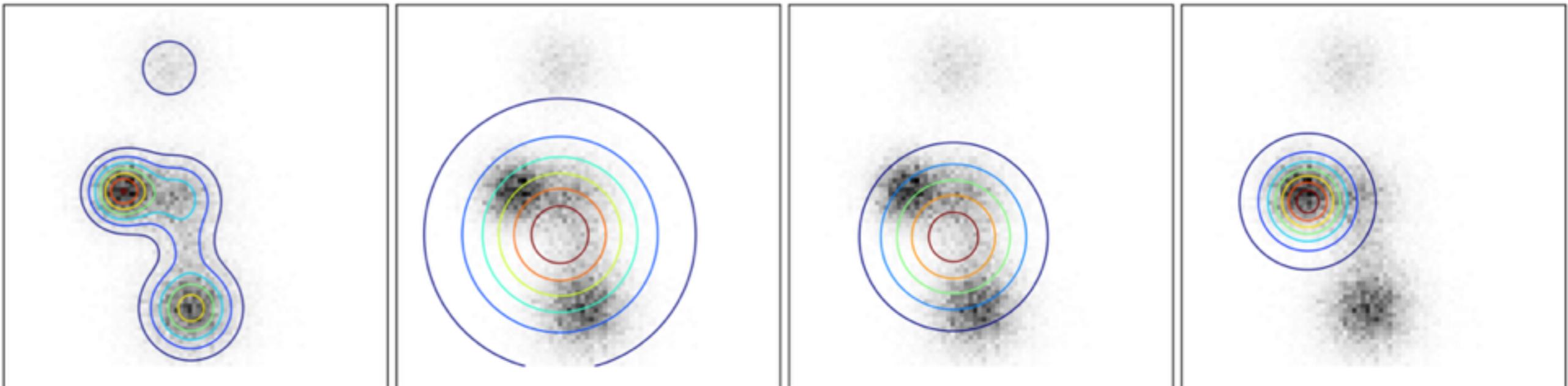
Discussion / Questions

A: P

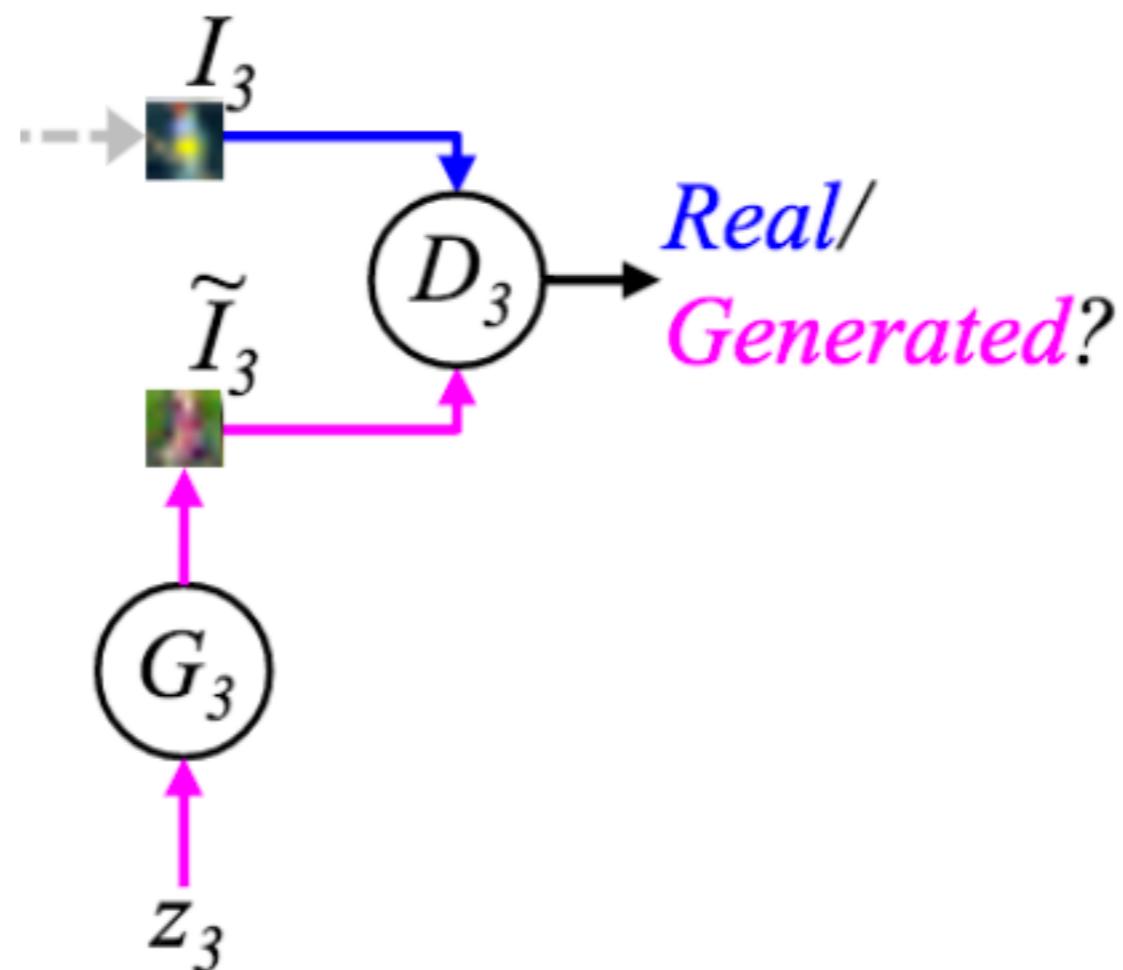
B: $\arg \min_Q JS_{0.1}[P\|Q]$

C: $\arg \min_Q JS_{0.5}[P\|Q]$

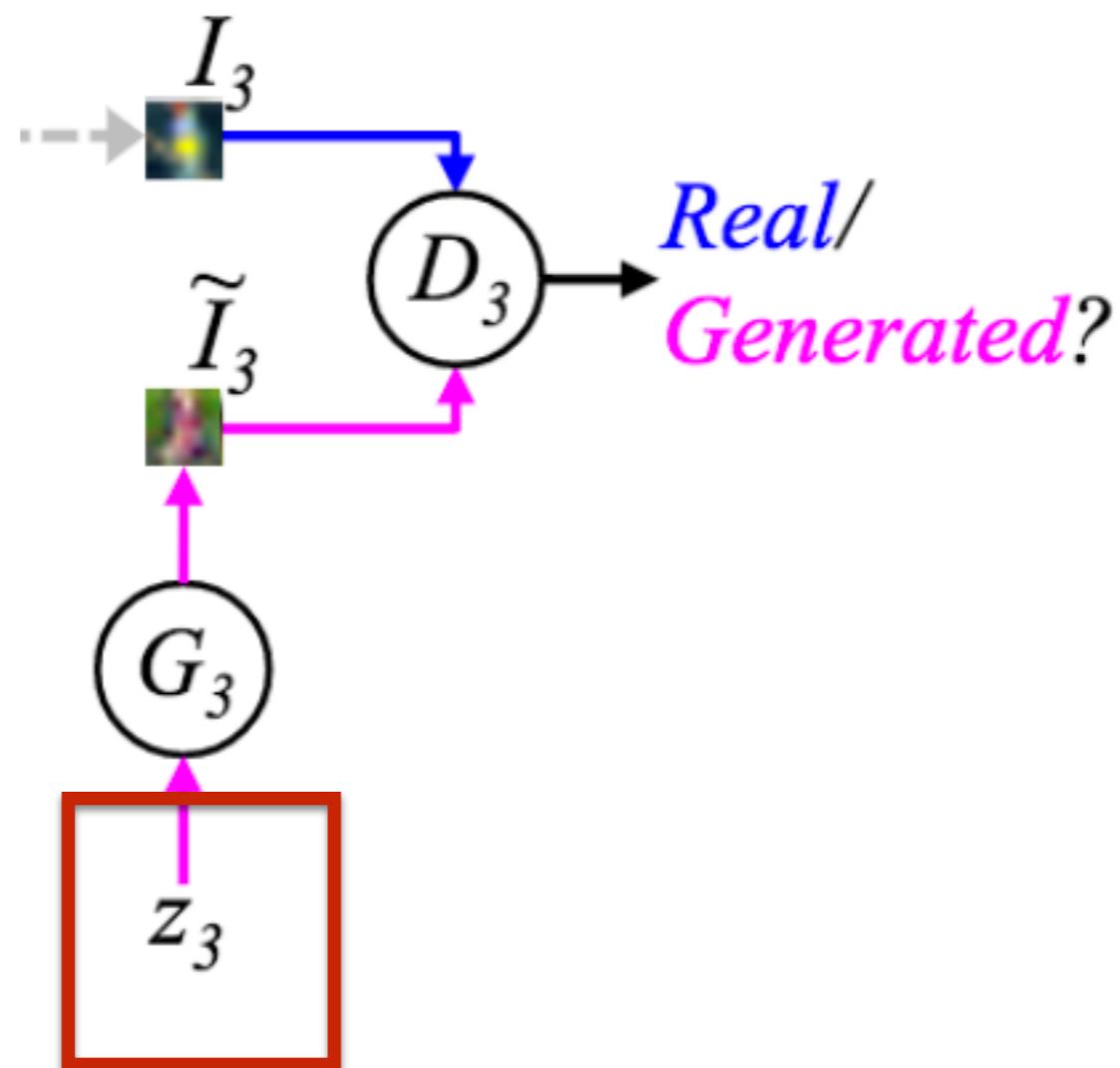
D: $\arg \min_Q JS_{0.99}[P\|Q]$



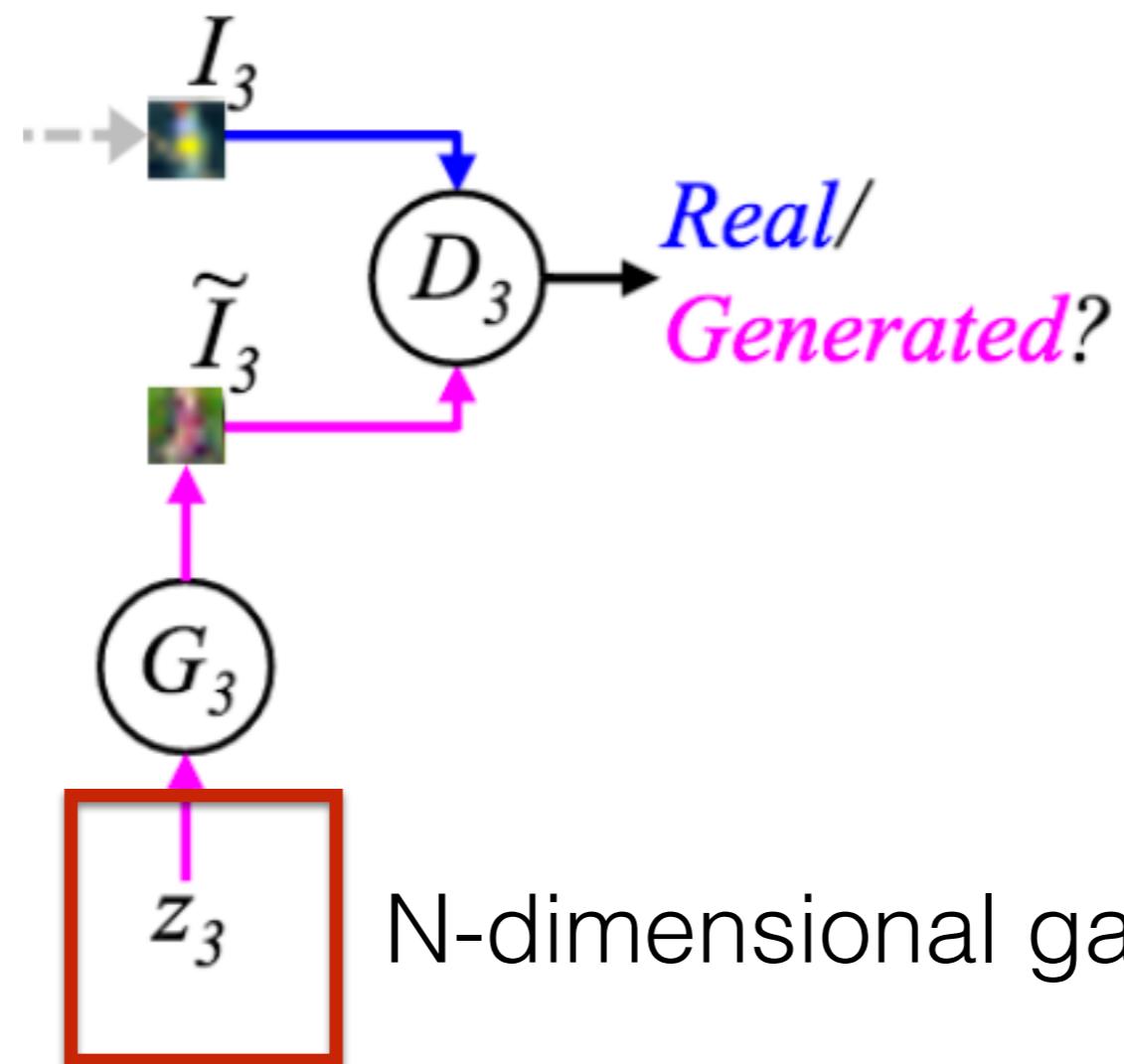
Structured Latent Distributions



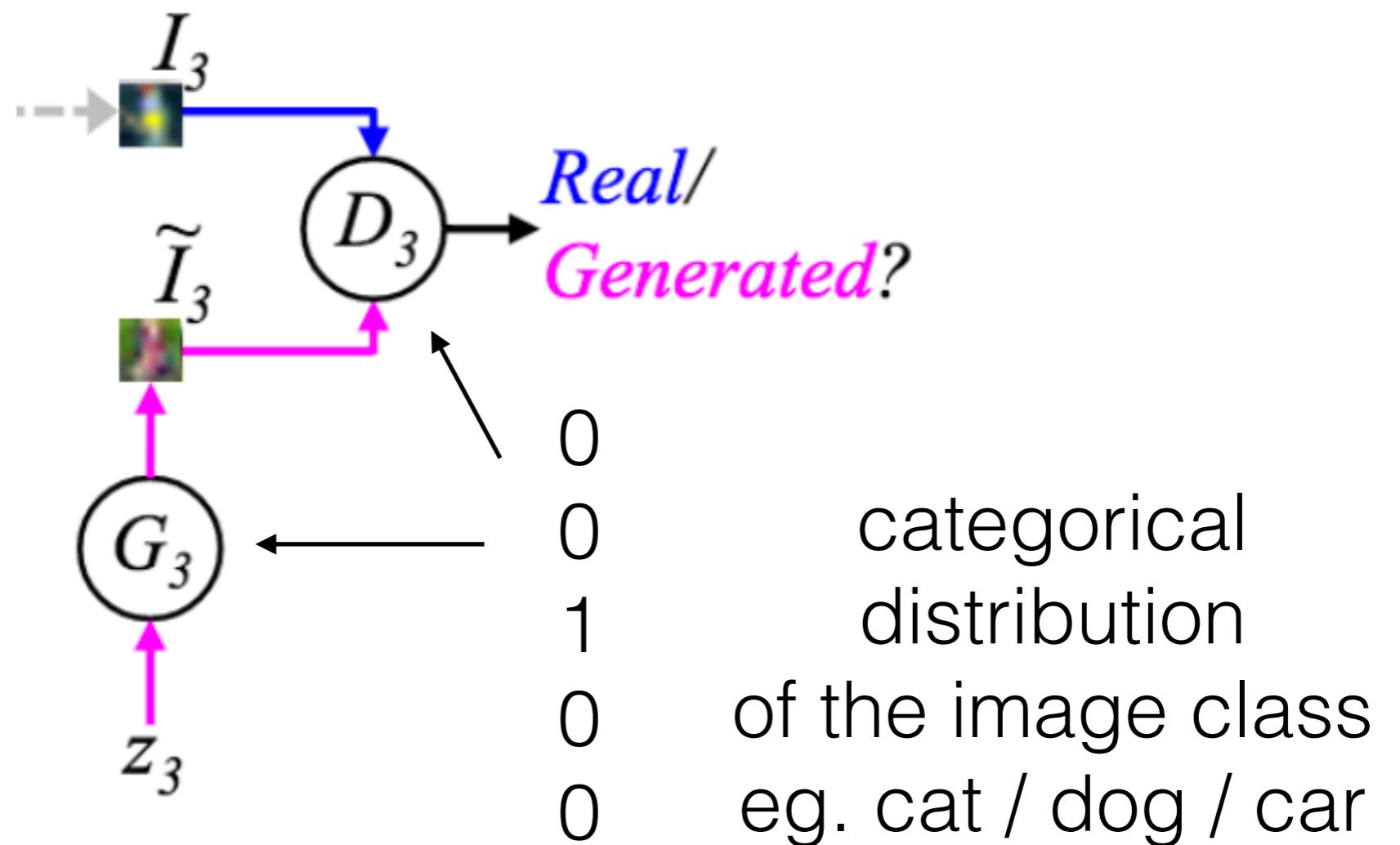
Structured Latent Distributions

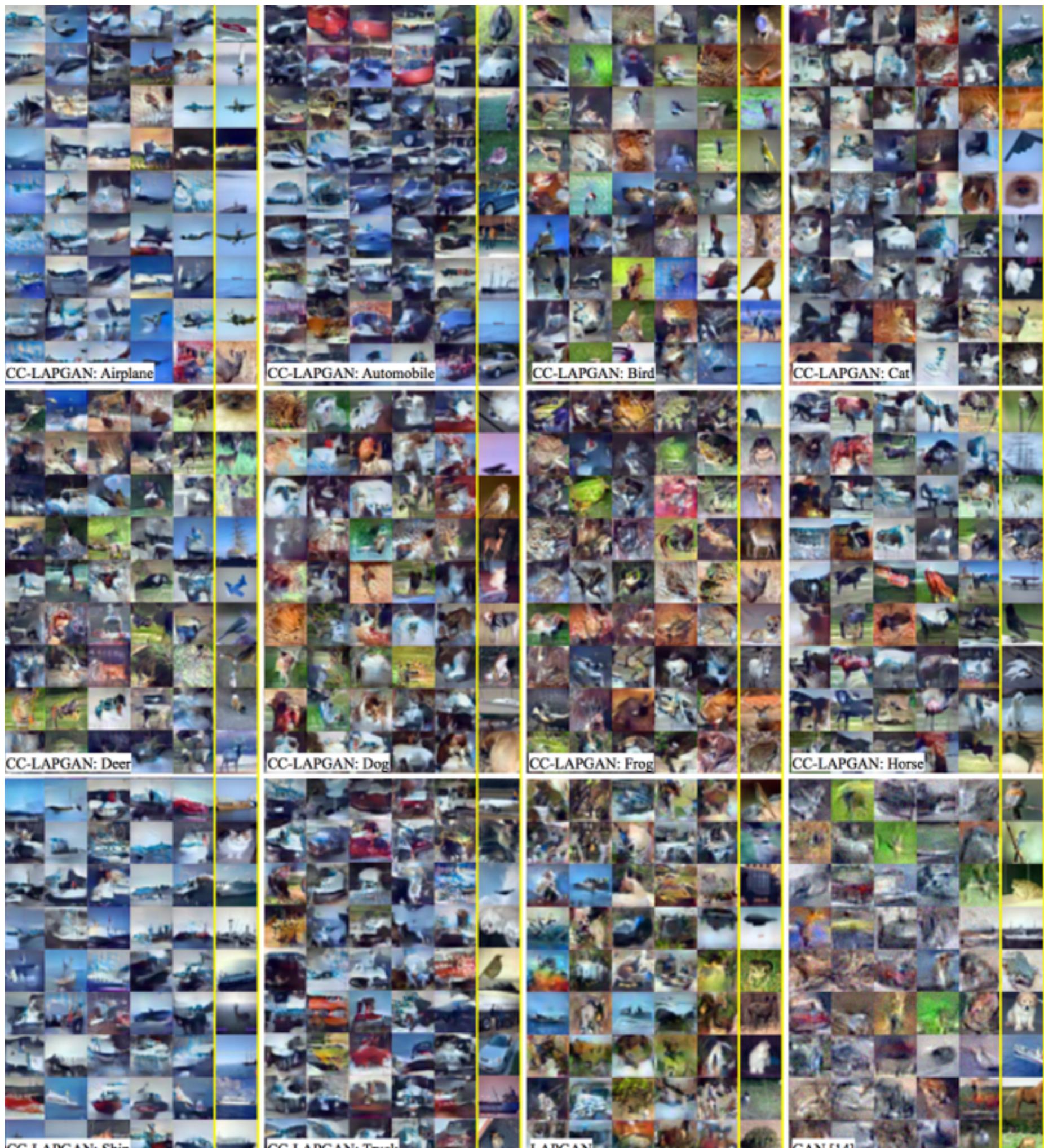


Structured Latent Distributions

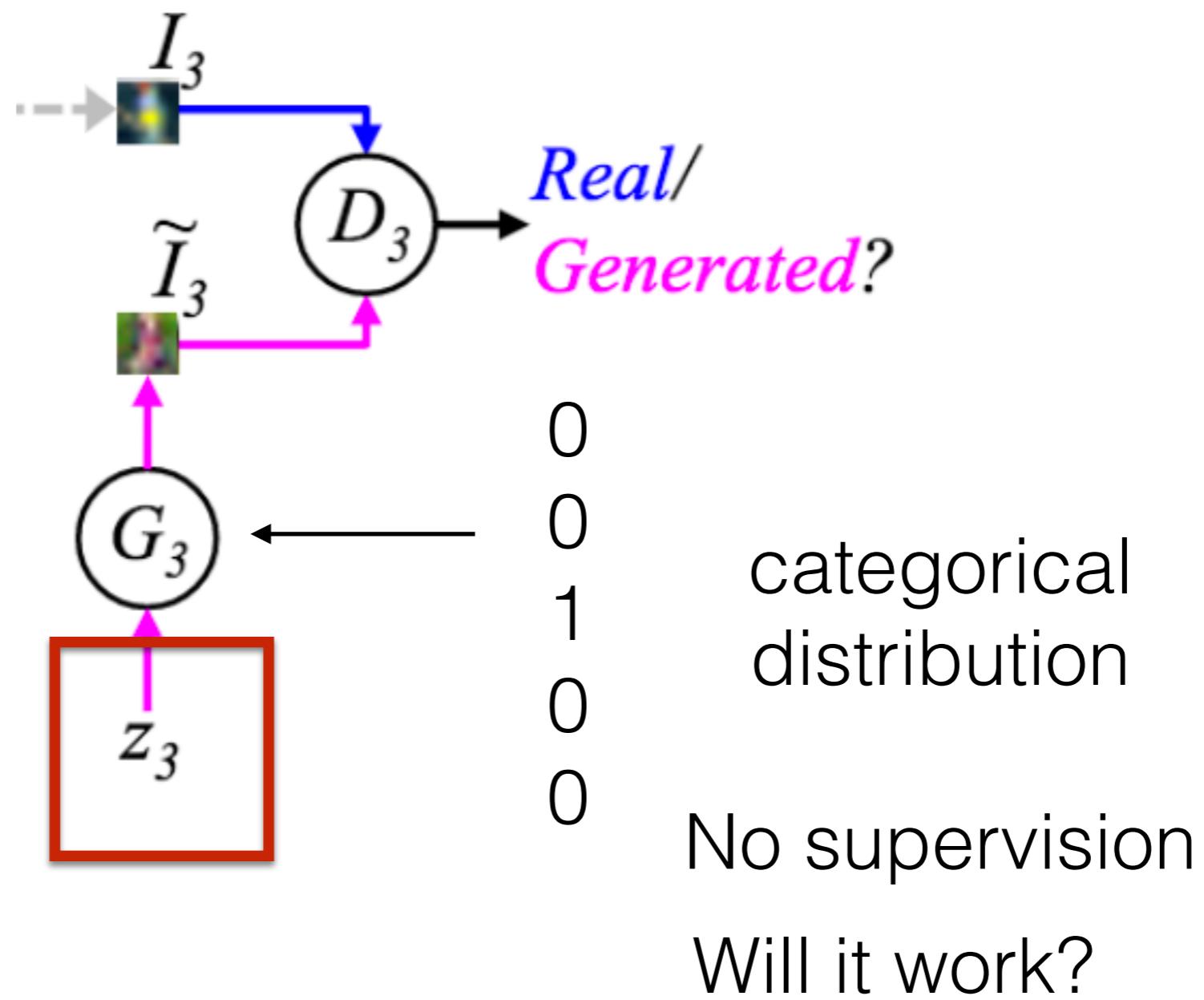


Structured Latent Distributions





Structured Latent Distributions



Discussion / Questions

