

10707

Deep Learning: Spring 2021

Andrej Risteski

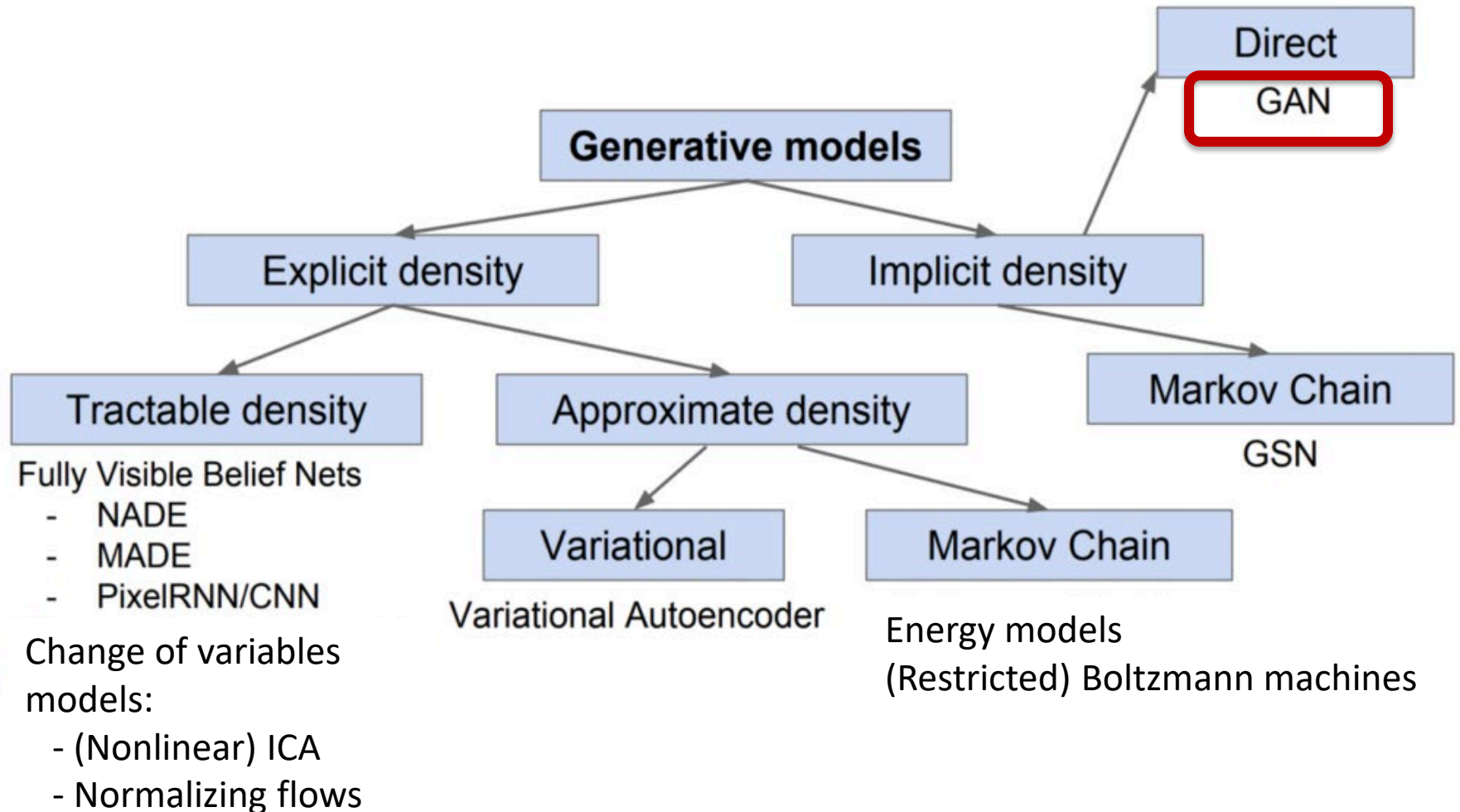
Machine Learning Department

Lecture 14:

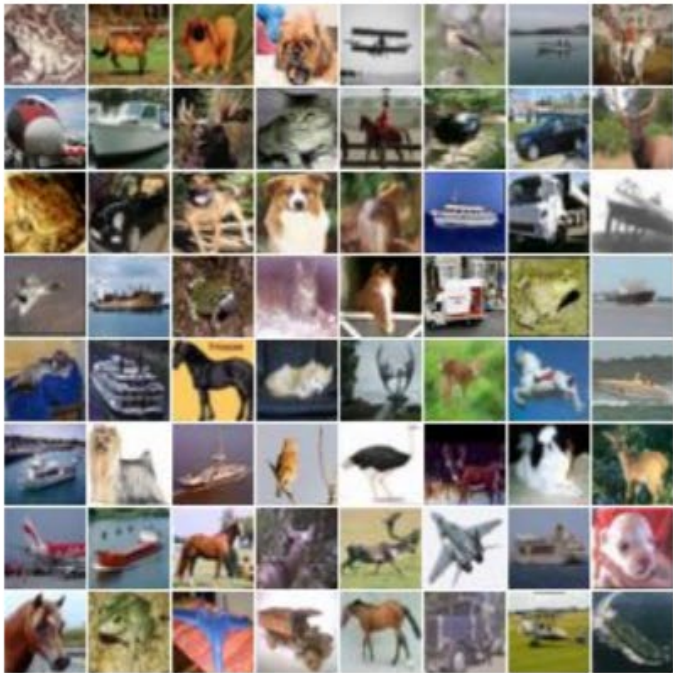
Implicit density models:
generative adversarial networks

The distribution $p_{\theta}(x)$ does not have an explicit form, but we can draw samples from it. e.g.

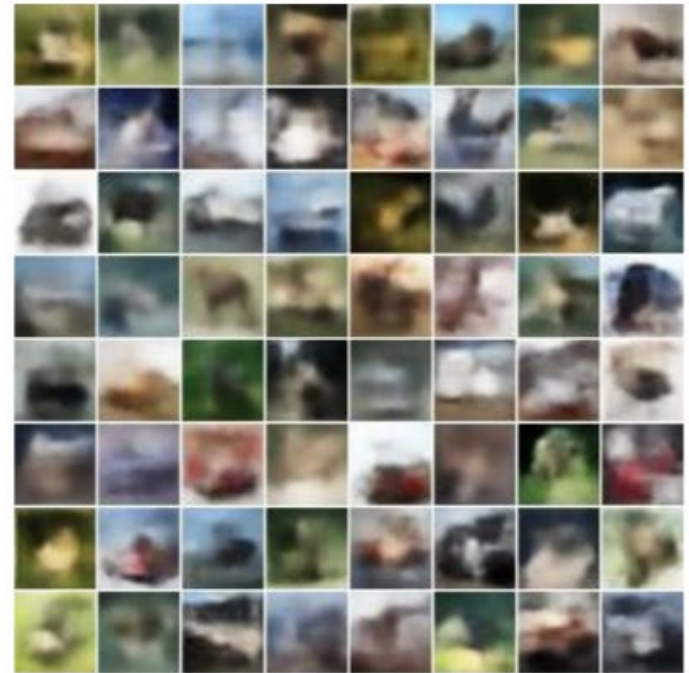
$$z \sim N(0, I); \quad x = G_{\theta}(z)$$



Some samples generated with VAEs and RBMs



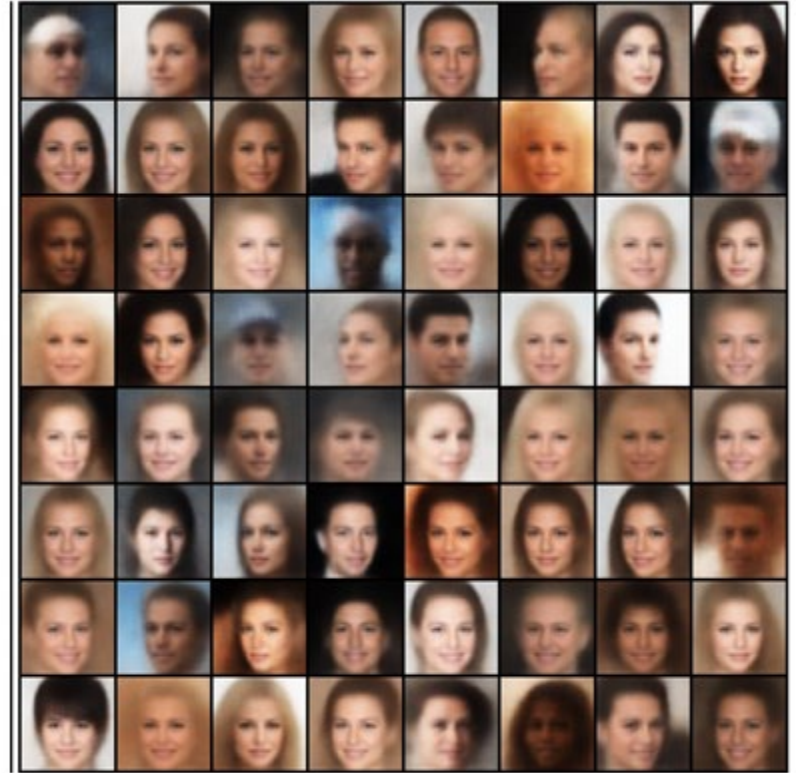
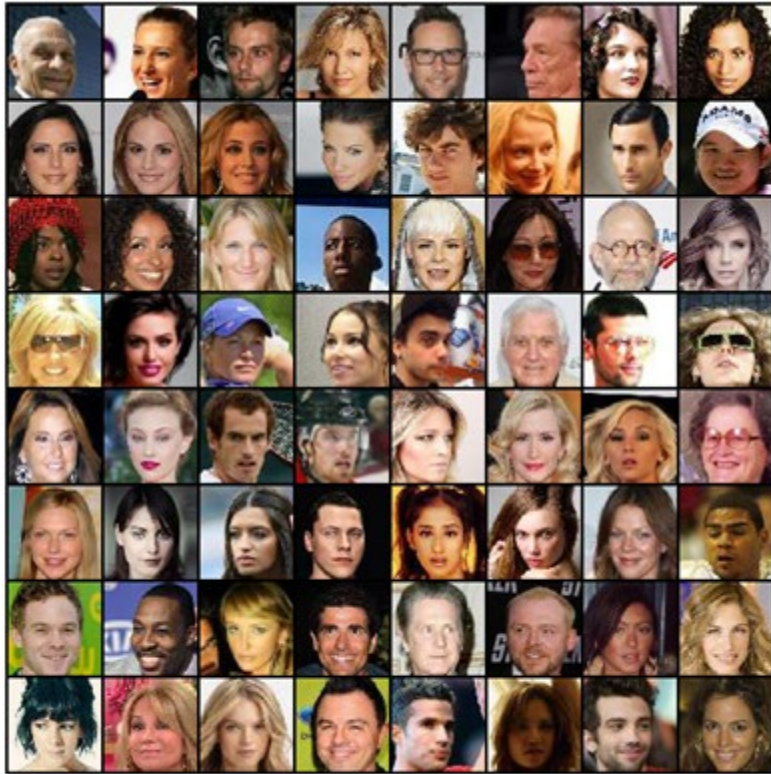
Data



VAE samples

Faces generated using a trained VAE, slides from
http://efrosgans.eecs.berkeley.edu/CVPR18_slides/VAE_GANS_by_Rosca.pdf

Some samples generated with VAEs and RBMs



Faces generated using a trained VAE

The problem

Samples are blurry, though they capture some high-level structure.

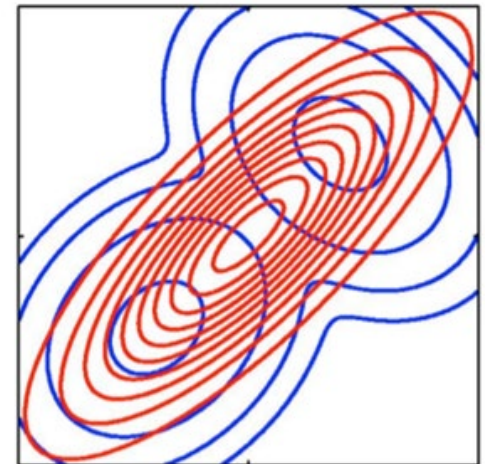
Some hypotheses for what goes wrong:

Strong metric: VAEs try to match the input distribution in KL divergence, which is quite a strong metric.

Poor posteriors: The posteriors in a VAE are Gaussian – very poor modeling power, e.g. cannot model multimodal distributions.

Max-likelihood encourages averaging:
finding the max-likelihood q to fit a distribution p is equivalent to minimizing $KL(p||q)$ (by expanding the def. of $KL = E_p \log p - E_p \log q$).

Recall from when we talked about variational methods: this KL tends to “average” modes.



$KL(p||q)$

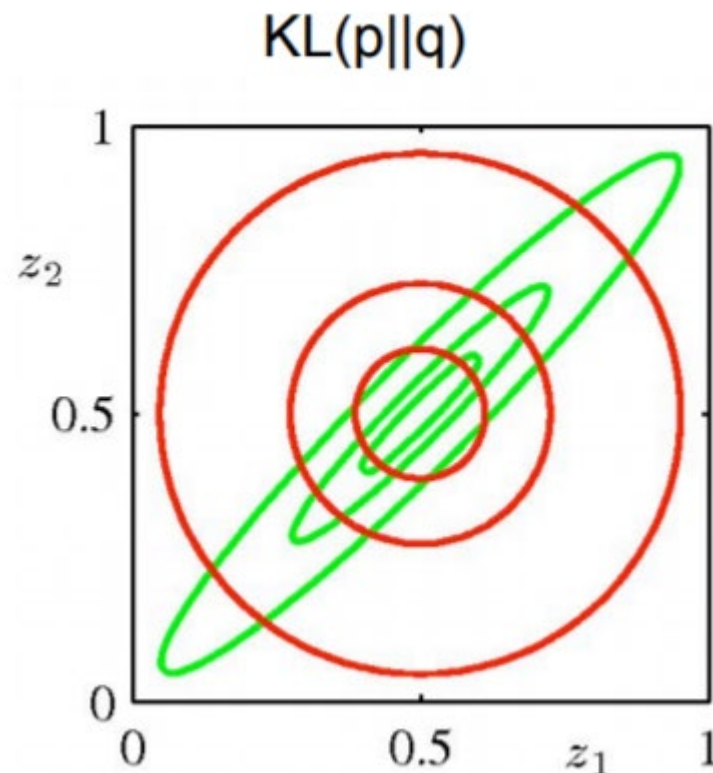
Recap: why $\text{KL}(p||q)$ averages

$$\text{KL}(p||q) = - \int p(\mathbf{Z}) \ln \frac{q(\mathbf{Z})}{p(\mathbf{Z})} d\mathbf{Z}.$$

There is a large positive contribution to the KL divergence from regions of \mathbf{Z} space in which:

- $q(\mathbf{Z})$ is near zero,
- unless $p(\mathbf{Z})$ is also close to zero.

Minimizing $\text{KL}(p||q)$ leads to distributions $q(\mathbf{Z})$ that **are nonzero in regions where $p(\mathbf{Z})$ is nonzero.**



The idea behind GANs

Matching a distribution on images is hard because we don't have good measures of “**distance**” between images. (Intuitively, two images could be very different in pixel space, while “**semantically**” being the same image.)

Why don't we simultaneously train a “distance metric” as we are training the model?

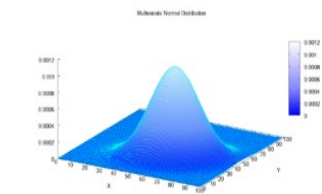
As a result, we will no longer be fitting the “maximum likelihood” model, but instead trying to learn some distribution close to the distribution of the input images in a learned “metric”.

This is (one of many) models which are “**likelihood-free**”: we won't be able to explicitly write a likelihood for the model, but (importantly) we will efficiently be able to draw samples from the model!

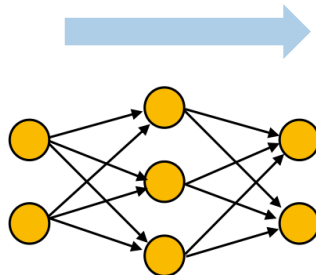
The GAN paradigm (Goodfellow et al. '14)

Goal: **Learn** a distribution close to some distribution we have few samples from. (Additionally, we will be able to sample efficiently from distribution.)

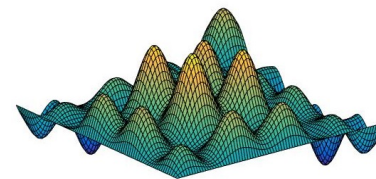
Approach: **Fit** distribution P_g parametrized by **neural network g**



$$Z \sim N(0, I_{k \times k})$$



Neural network $g(\cdot)$



$$X = g(Z)$$

The GAN paradigm (Goodfellow et al. '14)

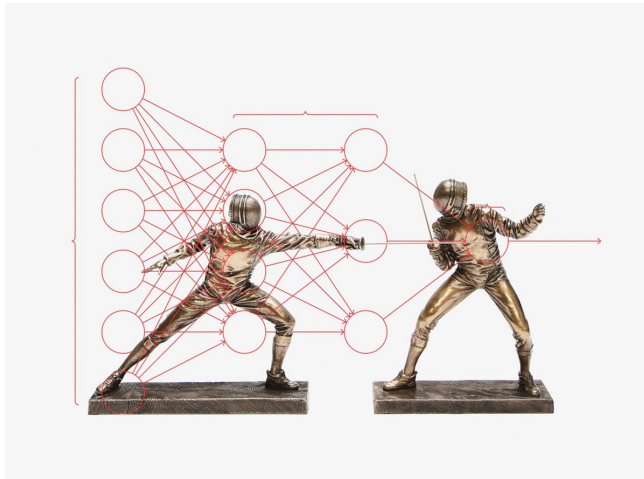
Photorealistic image/video generation

MIT
Technology
Review

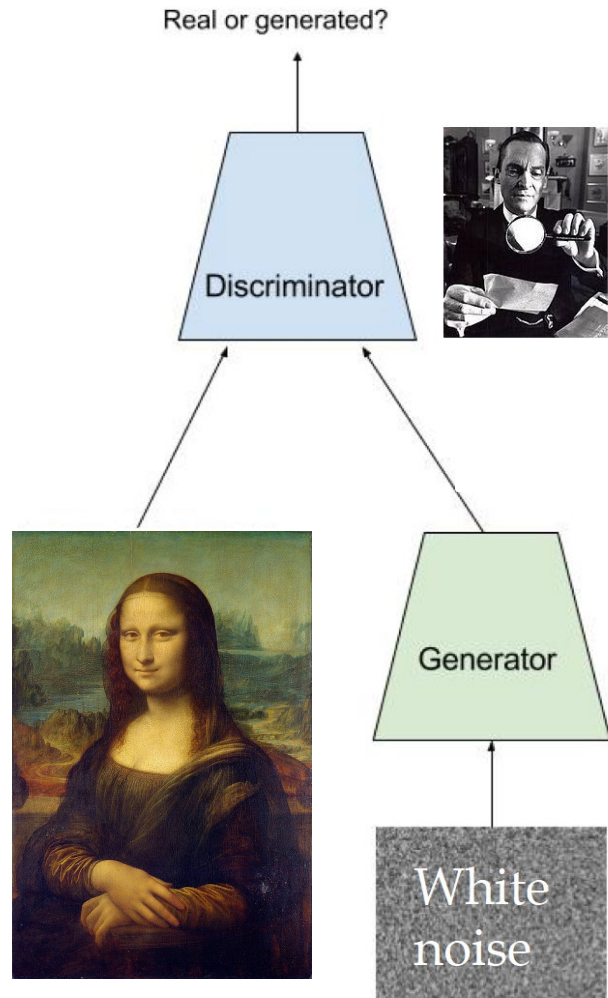
Top 10 Breakthrough
Technologies 2018



Extracting complex features



The GAN paradigm (Goodfellow et al. '14)



Game theoretic idea:

Generator trained to **fool** discriminator.

Discriminator trained to **beat** generator.



W-GAN formalization (Arjovsky et al. '17)

Min-max problem:

- ⊗ Min-player: generators $g \in G$; Max-player: discriminators $f \in F$.
- ⊗ Samples from image distr. P_{real} . Unif. distribution over samples: $P_{samples}$
- ⊗ P_g - generator distribution: $Z \sim N(0, I) \rightarrow g(Z)$

Training loss:

$$\min_{g \in G} \max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{samples}}[f] \right|$$

Difference of expectation of f
on **samples vs generated**
images



W-GAN formalization (Arjovsky et al. '17)

Min-max problem:

- ⌘ Min-player: generators $g \in G$; Max-player: discriminators $f \in F$.
- ⌘ Samples from image distr. P_{real} . Unif. distribution over samples: $P_{samples}$
- ⌘ P_g - generator distribution: $Z \sim N(0, I) \rightarrow g(Z)$

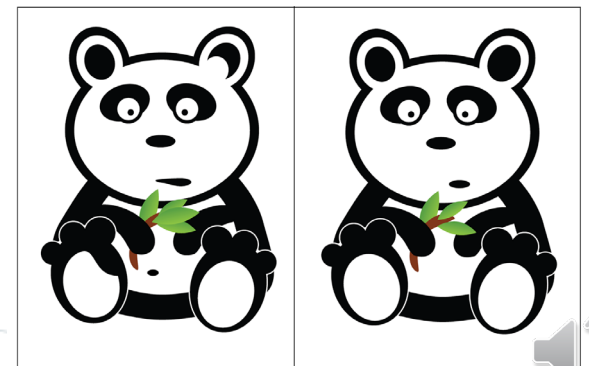
Training loss:

$$\min_{g \in G} \max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{samples}}[f] \right|$$

Generator g **fools**
discriminators F :

$$\forall f \in F, \mathbb{E}_{P_g}[f] \approx \mathbb{E}_{P_{samples}}[f]$$

Equivalently, small training
loss!



W-GAN formalization (Arjovsky et al. '17)

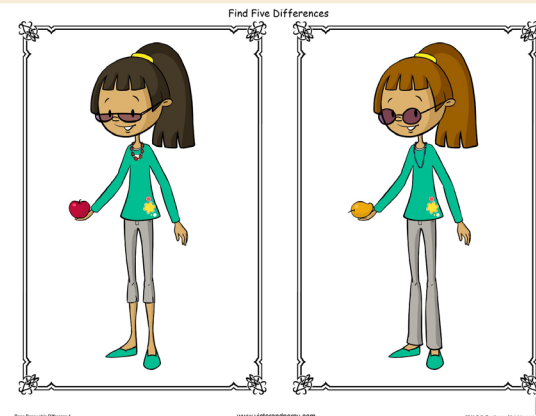
Min-max problem:

- ⊗ Min-player: generators $g \in G$; Max-player: discriminators $f \in F$.
- ⊗ Samples from image distr. P_{real} . Unif. distribution over samples: $P_{samples}$
- ⊗ P_g - generator distribution: $Z \sim N(0, I) \rightarrow g(Z)$

Training loss:

$$\min_{g \in G} \max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{samples}}[f] \right|$$

Discriminators F **beat** generators if for all $g \in G$, there is an $f \in F$ such that $\mathbb{E}_{P_g}[f] \neq \mathbb{E}_{P_{samples}}[f]$



W-GAN formalization (Arjovsky et al. '17)

Min-max problem:

- ⌘ Min-player: generators $g \in G$; Max-player: discriminators $f \in F$.
- ⌘ Samples from image distr. P_{real} . Unif. distribution over samples: $P_{samples}$
- ⌘ P_g - generator distribution: $Z \sim N(0, I) \rightarrow g(Z)$

Training loss:

$$\min_{g \in G} \max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{samples}}[f] \right|$$

$$d_F(P_{samples}, P_g)$$

Discriminators F **beat** generators if for all $g \in G$, there is an $f \in F$ such that $\mathbb{E}_{P_g}[f] \neq \mathbb{E}_{P_{samples}}[f]$

“**Distance**” specified by discriminators F .
Captures how well F ’s can **distinguish** two distributions

W-GAN formalization (Arjovsky et al. '17)

Min-max problem:

- ⌘ Min-player: generators $g \in G$; Max-player: discriminators $f \in F$.
- ⌘ Samples from image distr. P_{real} . Unif. distribution over samples: $P_{samples}$
- ⌘ P_g - generator distribution: $Z \sim N(0, I) \rightarrow g(Z)$

Training loss:

$$\min_{g \in G} \max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{samples}}[f] \right|$$

$$d_F(P_{samples}, P_g)$$

Discriminators F **beat** generators if for all $g \in G$,
there is an $f \in F$
 $\mathbb{E}_{P_g}[f] \not\approx \mathbb{E}_{P_{samples}}[f]$

$$\text{Training loss} = \min_{g \in G} d_F(P_g, P_{samples})$$

Examples of distances d_F

$$\max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{\text{samples}}}[f] \right|$$

$$d_F(P_{\text{samples}}, P_g)$$

Absolute value
can be removed
(-f is Lip if f is Lip)

$F = \{f: |f|_\infty \leq 1\}$: **Total variation distance**

Measures differences of bounded functions

$F = \{f: \text{Lip}(f) \leq 1\}$: **W_1 (Wasserstein, earthmover) distance**

Measures differences of 1-Lipschitz functions



Examples of distances d_F

$$\max_{f \in F} \left| \mathbb{E}_{P_g}[f] - \mathbb{E}_{P_{\text{samples}}}[f] \right|$$

$$d_F(P_{\text{samples}}, P_g)$$

If distance d_F is a **metric**: $d_F(p, q) \geq 0$ and $d_F(p, q) = 0$ only if $p = q$

Hence, if we learn a distribution P_g , s.t. $d_F(P_g, P_{\text{real}}) = 0$, and P_{real} is the true data distribution, we have $P_g = P_{\text{real}}$.

In the limit of infinite samples, $P_{\text{real}} = P_{\text{samples}}$, so if training error is 0, we have learned a distribution $P_g = P_{\text{real}}$



Variants

Monotone function ϕ

$$\max_{f \in F} \mathbb{E}_{P_g}[\phi(f)] - \mathbb{E}_{P_{\text{samples}}}[\phi(1 - f)]$$

$$d_F(P_{\text{samples}}, P_g)$$

Maximize log-probability
of correct answer

$\phi = \log$: DC-GAN

$$\max_{f \in F} \mathbb{E}_{P_g}[\log(f)] - \mathbb{E}_{P_{\text{samples}}}[\log(1 - f)]$$

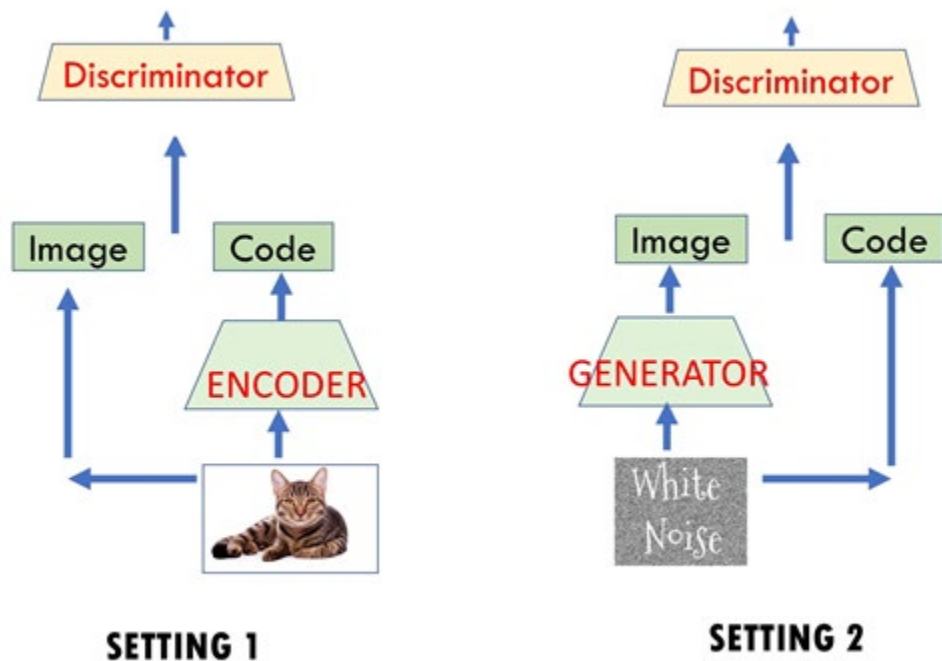
If F contains all function $f: \mathbb{R}^d \rightarrow [0,1]$, the above objective converges to

$$\min_{g \in G} \underbrace{KL(P_g || (P_{\text{real}} + P_g)/2) + KL((P_{\text{real}} + P_g)/2 || P_g)}_{\text{Jensen-Shannon divergence}} - 2 \ln 2$$

Jensen-Shannon divergence

Variants

If we also want to train an **encoder E**: that is, a network that tries to output the z , s.t. x was generated from z , there is a way to adapt the adversarial setup:



Discriminator tries to distinguish b/w:

Setting 1: samples are $(x, E(x))$

Setting 2: samples are $(z, G(z))$

Dumoulin et al, Donahue et al '16:

In the limit of infinite samples, infinite capacity generators, the distributions of

Setting 1 and 2 match.

What affects our choice of F ?

Statistical considerations: very powerful discriminators (e.g. large neural networks) will require a lot of samples. Weak discriminators will specify a very weak metric: very “different” distributions will look very “similar” to metric.

Our understanding here is much better.

Algorithmic considerations: if discriminators are very powerful, gradient information for generator is too weak and can vanish. If they are too weak – metric is weak.

Our understanding of training dynamics is very poor.



Statistical questions

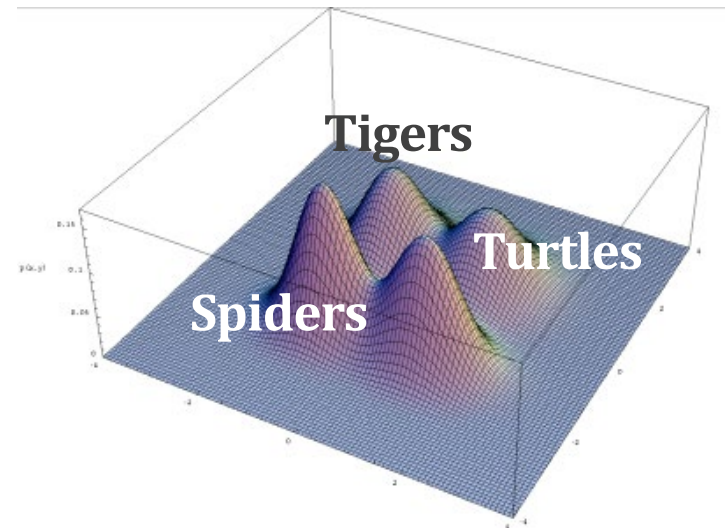


Tension: strength of discriminators?

Small (weak) discriminators \Rightarrow mode collapse:

Neural net discriminators with $\leq m$ parameters
fooled by generator w/ support size $\approx m$.

[Arora et al'17, Arora-Risteski-Zhang ICLR'18]



Real-life distributions
have large support!

Tension: strength of discriminators?

Small (weak) discriminators \Rightarrow mode collapse.

Happens for any P_{real}

Neural net discriminators with $\leq m$ parameters
fooled by generator w/ support size $\approx m$.

[Arora et al'17, Arora-Risteski-Zhang ICLR'18]

Not memorization!
More training samples
don't help.



Discriminators too weak: d_F cannot distinguish between small-support distr. and P_{real} .

Real-life distributions
have large support!

Tension: strength of discriminators

Small discriminators \Rightarrow mode collapse:

Generator w/ support size $\approx m$ **fools**
neural net discriminators with $\leq m$ **parameters**.
[Arora et al'17, Arora-Risteski-Zhang 'ICLR18]

Large discriminators \Rightarrow poor generalization:

Loss with small # samples differs a lot from loss with infinite # samples.

$$d_F(P_{\text{samples}}, P_g) \not\approx d_F(P_{\text{real}}, P_g)$$

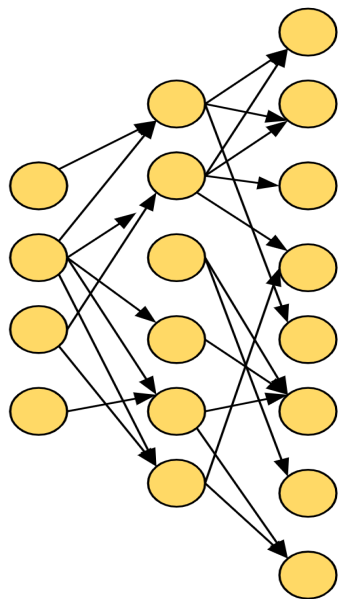


Sweet spot for natural distributions

Let P_{real} itself be generated by neural net. ($P_{real} = P_g, g \in G$)

Let $G = \{ \text{1-to-1 neural networks of bounded size} \}$

Design **small** discriminators F w/ good distinguishing power.



⌘ Less general than arbitrary neural-net generators

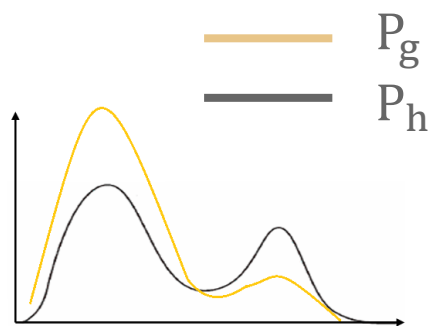
⌘ Allows data to lie on **low-dim. manifold**.



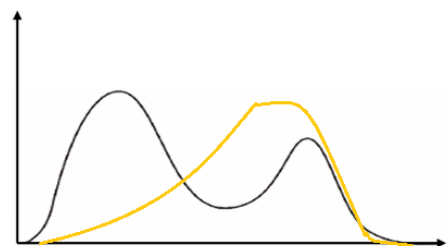
Distinguishing power

Discriminators F have **distinguishing power** against generators

G , if: $\forall g, h \in G : d_F(P_g, P_h) \gtrsim W_1(P_g, P_h)$

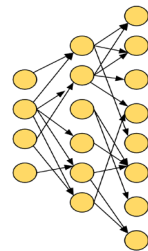


Distance d_F is not too much weaker than W_1 , **only** for distributions in our class.



Main

Neural nets of slightly larger
depth/size than generators.
(Suggestion for practice!)



Thm (Bai-Ma-Risteski ICLR 19): **Small** discriminators F with **distinguishing power** for $G = \{1\text{-to-1 neural nets of bdd size}\}$ exist.

So, if P_{real} generated by 1-to-1 neural net with d params,

w/ **poly(d)** samples, $d_F(P_{samples}, P_g) \leq \epsilon \Rightarrow W_1(P_{real}, P_g) \leq O(\sqrt{\epsilon})$

Training was successful

True distribution learned.



Algorithmic questions



How to train a GAN

“**Best response dynamics**”: fix generator, find best discriminator; then fix discriminator, find best generator. Repeat.

Better in practice: take one gradient step for generator, do a few gradient steps for discriminator. Repeat.

Going with intuition of Wasserstein distance: we'd like the discriminators to be somewhat Lipschitz – **clipping** weights is a good idea.



How to train a GAN

How many discriminator gradient steps to take for each generator gradient step

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α , the learning rate. c , the clipping parameter. m , the batch size.
 n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

Empirical estimates of expectations to calculate discriminator gradient

Clip

Generator gradient

Figure from Arjovsky, Chintala, Bottou '17

Common training problems

Unstable training: the problem is a min-max problem (also called saddle point problem) – typically optimization is much less stable than pure minimization.

Particularly common instantiation: **cycling**

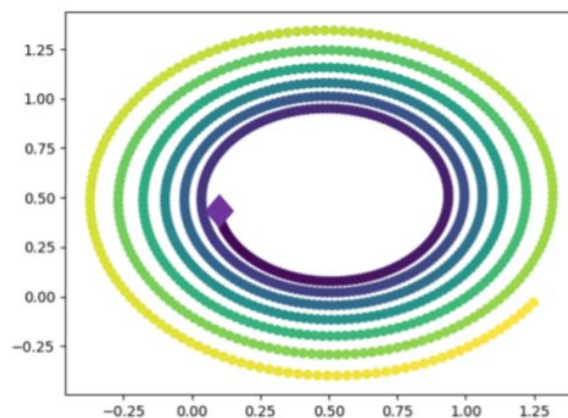


Figure 5: $f(x,y)=(x-1/2)(y-1/2)$. x and y are initialized at the purple diamond. Alternating between gradient ascent/ descent on x and y leads to divergent behavior, spiraling away from the optimum, but the average of the parameters is close to the optimal solution

Figure from https://people.csail.mit.edu/madry/6.883/files/lecture_8.pdf

Common training problems

Unstable training: the problem is a min-max problem (also called saddle point problem) – typically optimization is much less stable than pure minimization.

Vanishing gradient: if the discriminator is too good, the generator gradients have a propensity to be small. (This is concerning, as to be taking gradients of the Wasserstein/JS/... objective, the discriminator needs to be optimal.)

Less of a problem with more modern GANs than with DC-GAN.

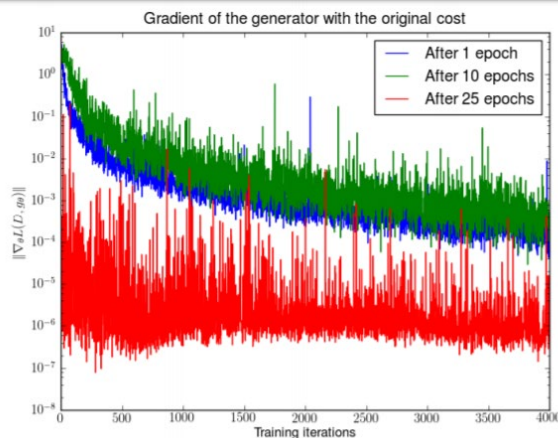


Figure 2: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the original cost function. We see the gradient norms decay quickly, in the best case 5 orders of magnitude after 4000 discriminator iterations. Note the logarithmic scale.

Figure from Arjovsky & Bottou '17

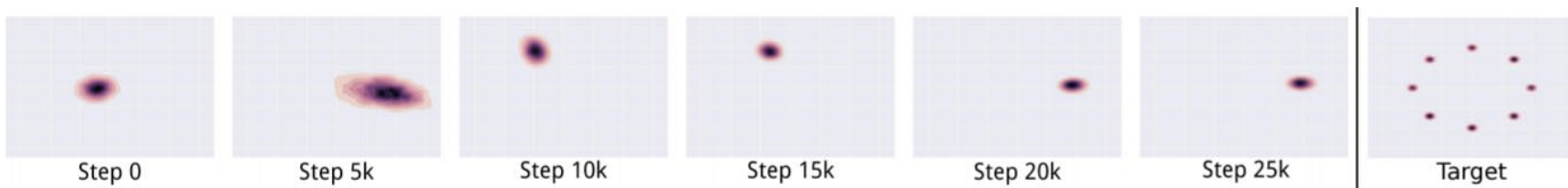
Common training problems

Unstable training: the problem is a min-max problem (also called saddle point problem) – typically optimization is much less stable than pure minimization.

Vanishing gradient: if the discriminator is too good, the generator gradients have a propensity to be small. (This is concerning, as to be taking gradients of the Wasserstein/JS/... objective, the discriminator needs to be optimal.)

Less of a problem with more modern GANs than with DC-GAN.

Mode collapse: the training only recovers some of the modes of the underlying distribution. (**NOT** clear if this is a statistical or algorithmic problem.)

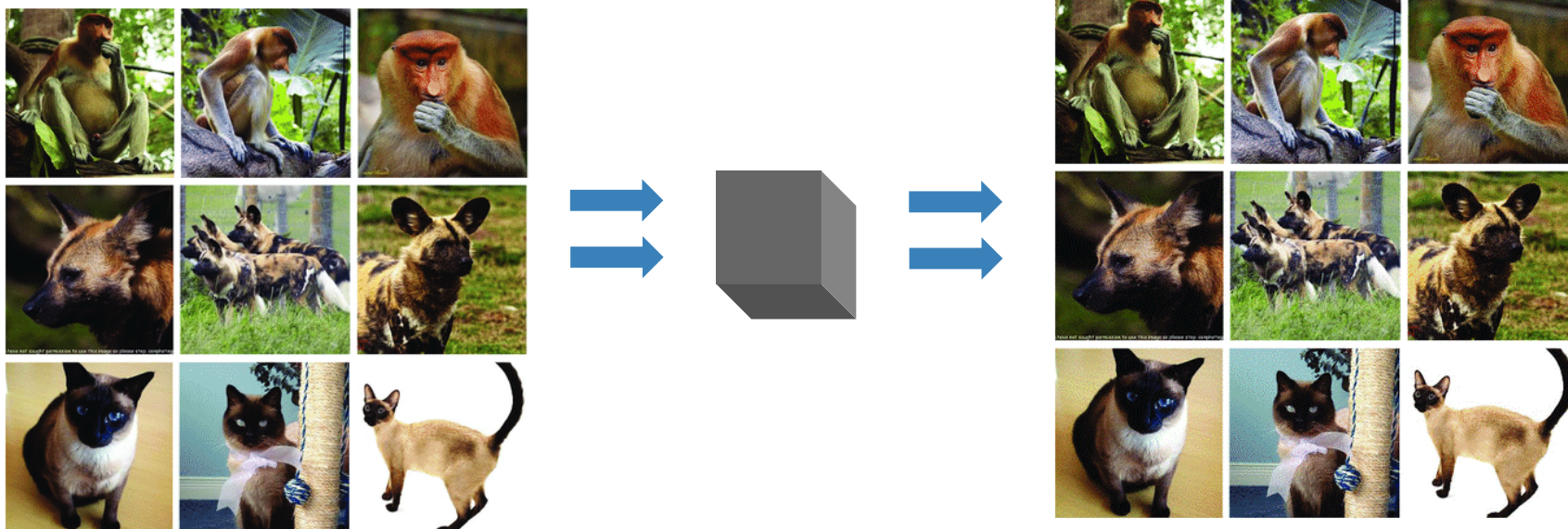


Evaluating GANs



How do we evaluate GANs

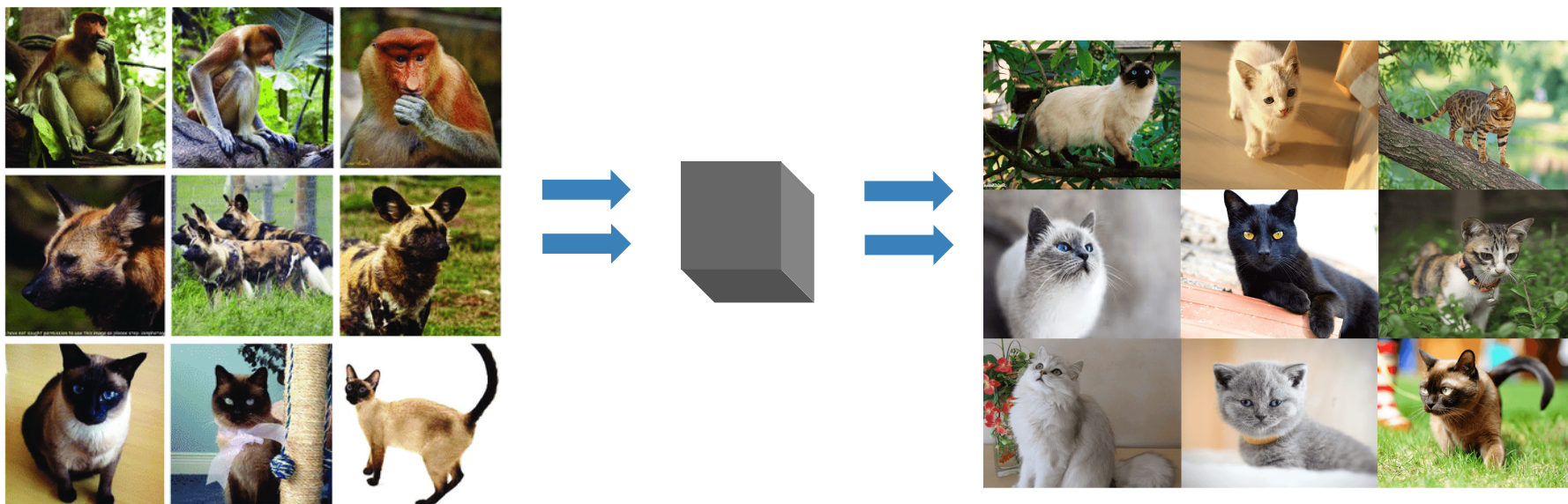
Wins beauty contest, but does the GAN really learn distribution?



No, just memorizing training samples.

How do we evaluate GANs

Wins beauty contest, but does the GAN really learn distribution?



No, mode collapse: missing regions (modes) of input distribution.

How do we evaluate GANs

Since we cannot evaluate the likelihood of the input data under a generator, **evaluation is hard**.

(Disproportionately) frequently, the evaluation is done by visually comparing samples – this cannot exclude issues like **memorization**, **mode collapse**, etc.

*Can we test for some common **failure modes**?*



Diagnosing small support size: bday paradox



Birthday Paradox:

If there are **23** people in a group, $> \frac{1}{2}$ chance that two of them share a birthday.

General version: Suppose a distribution is uniform over N images. Then

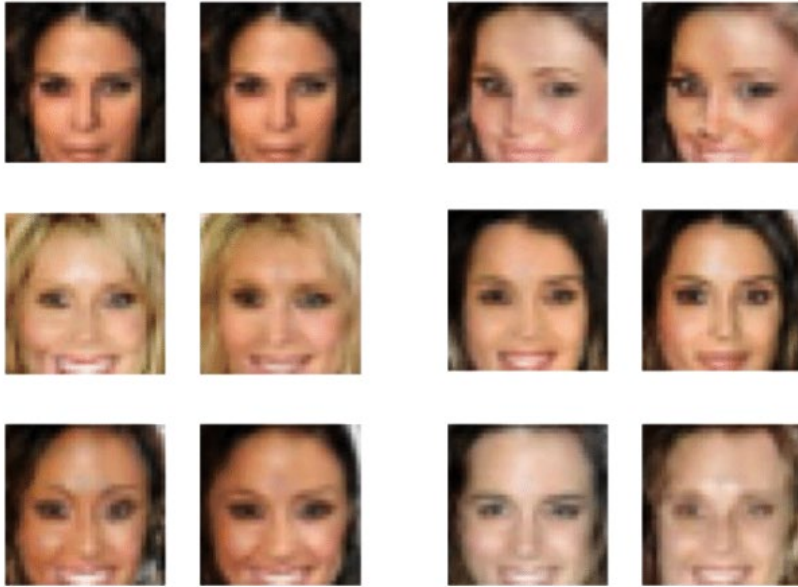
$\Pr[\text{sample of size } \sqrt{N} \text{ has a duplicate image}] > \frac{1}{2}.$

Birthday paradox test* [Arora-Risteski-Zhang ICLR'18] : If a sample of size s has **duplicate** images with prob. $> \frac{1}{2}$, then distribution essentially* has only **s^2 distinct images**.

Implementation: Draw sample of size s ; heuristically flag possible near-duplicates. Use human in the loop to verify duplicates.



Diagnosing small support size: bday paradox



CelebA (faces): 200k training images

DC-GAN [Radford et al.'15]:

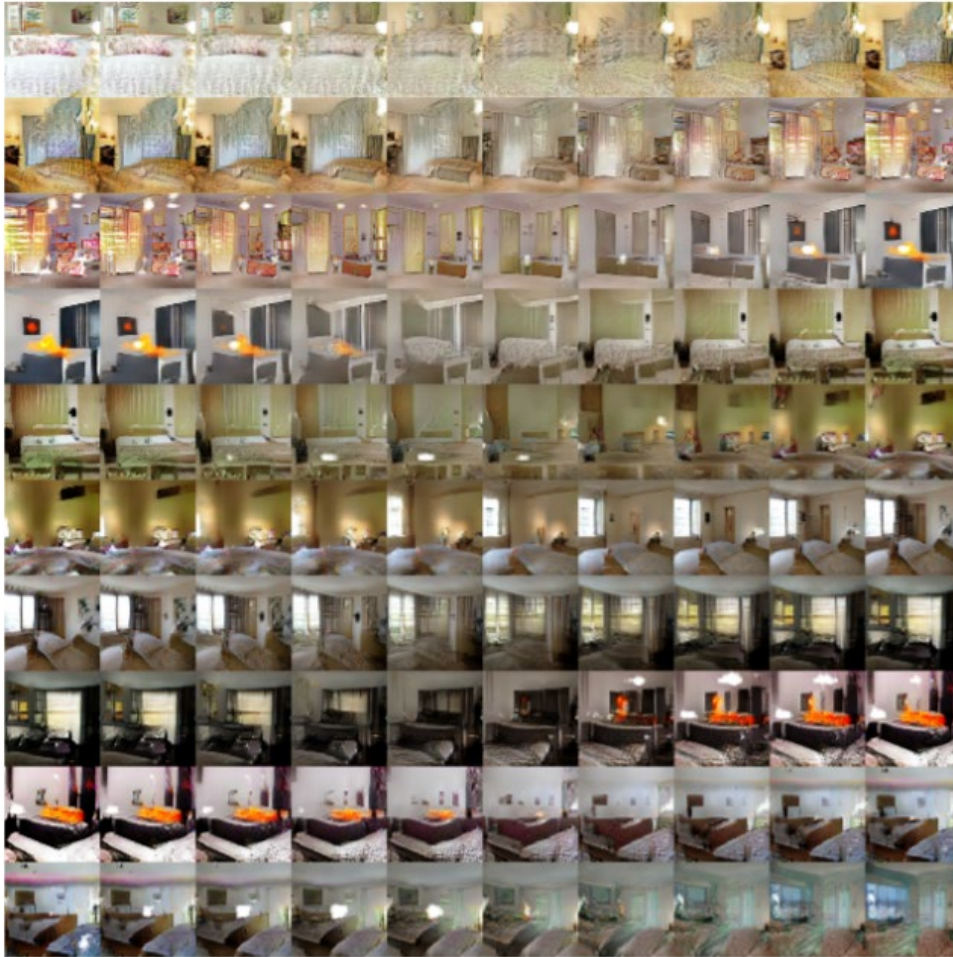
Support size $\approx 250K$

**BiGAN [Donohue et al.'17] and
ALI [Dumoulin et al.'17]:**

Support size $\approx 1M$

A lot of **followup** and **parallel** work about diagnosing mode collapse.

Interpolation



If linearly interpolating in latent space gives rise to meaningful images (without sharp transitions), unlikely GAN is just memorizing.

Figure from
Radford, Metz, Chintala '16.

Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

Inception score

Suppose we use trained network – the *Inception* architecture as a **labeler** for images. Inception gives probability over labels y for sample x : $p(y|x)$.

Desirable features of generator: the Inception classifier should be “sure” about the label for most images ($p(y|x)$ should have *low entropy*), and the classes it generates should be diverse ($p(y) = \mathbb{E}_{x \sim P_g} p(y|x)$ should have *high entropy*)

Thus, we want $H(p(y|x))$ to be **low**, $H(p(y))$ is **high**.

Consider the expression: $\mathbb{E}_{x \sim P_g} KL(p(y|x) || p(y))$

$$= \mathbb{E}_{x \sim P_g} \mathbb{E}_{y \sim p(y|x)} \log p(y|x) - \log p(y)$$

$$= -\mathbb{E}_{x \sim P_g} H(p(y|x)) + H(p(y))$$



Inception score

Suppose we use trained network – the *Inception* architecture as a **labeler** for images. Inception gives probability over labels y for sample x : $p(y|x)$.

Desirable features of generator: the Inception classifier should be “sure” about the label for most images ($p(y|x)$ should have *low entropy*), and the classes it generates should be diverse ($p(y) = \mathbb{E}_{x \sim P_g} p(y|x)$ should have *high entropy*)

Thus, we want $H(p(y|x))$ to be **low**, $H(p(y))$ is **high**.

Consider the expression: $\mathbb{E}_{x \sim P_g} KL(p(y|x) || p(y))$

Inception score: $\exp(\mathbb{E}_{x \sim P_g} KL(p(y|x) || p(y)))$



Inception score

Suppose we use trained network – the *Inception* architecture as a **labeler** for images. Inception gives probability over labels y for sample x : $p(y|x)$.

Desirable features of generator: the Inception classifier should be “sure” about the label for most images ($p(y|x)$ should have *low entropy*), and the classes it generates should be diverse ($p(y) = \mathbb{E}_{x \sim P_g} p(y|x)$ should have *high entropy*)

Many follow ups, e.g. Frechet inception distance, modified inception score, ...

Check for **many** other metrics:

Borji '18: 24 quantitative, 5 qualitative measures

