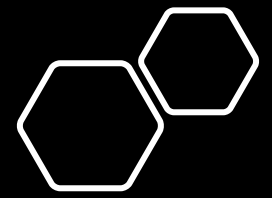
AIMS | African Institute for Mathematical Sciences RWANDA

AMMI Review sessions

Deep Learning (10)

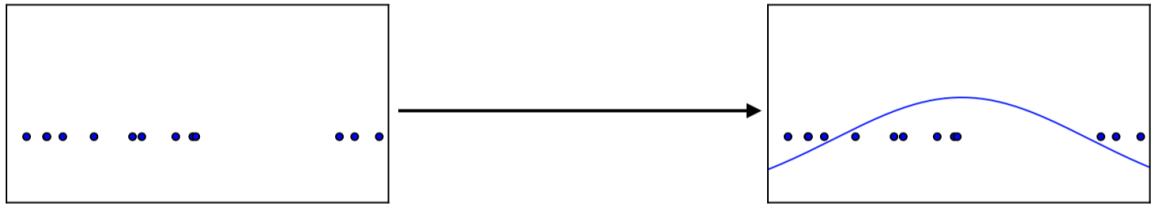
GANs



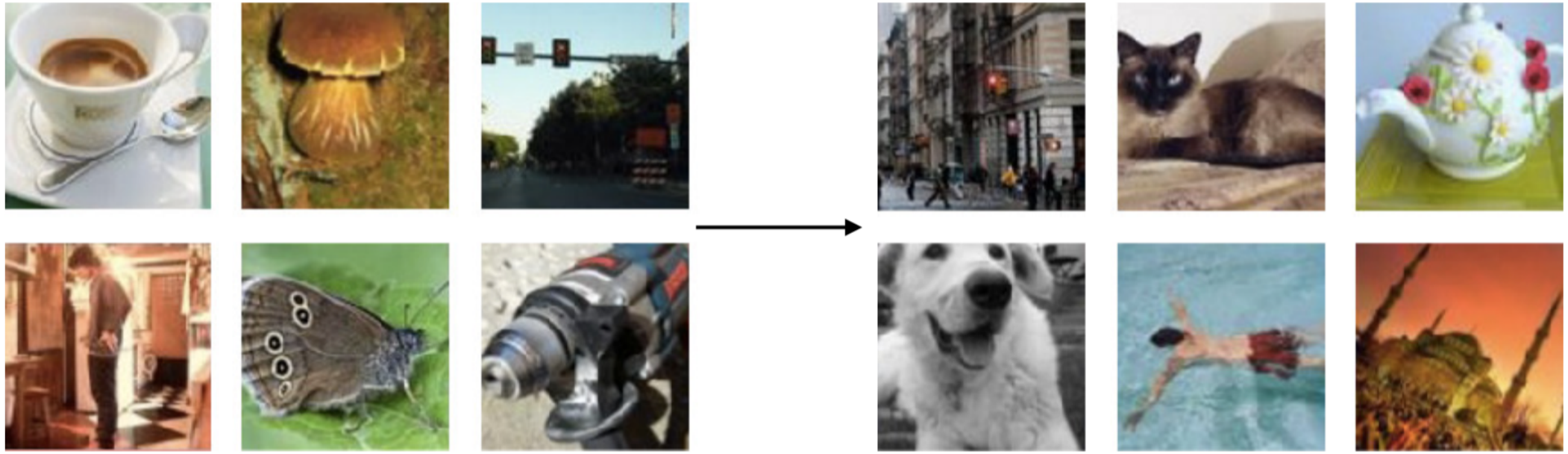
March 6th 2020

Generative Models

- Density estimation



- Samples Generation



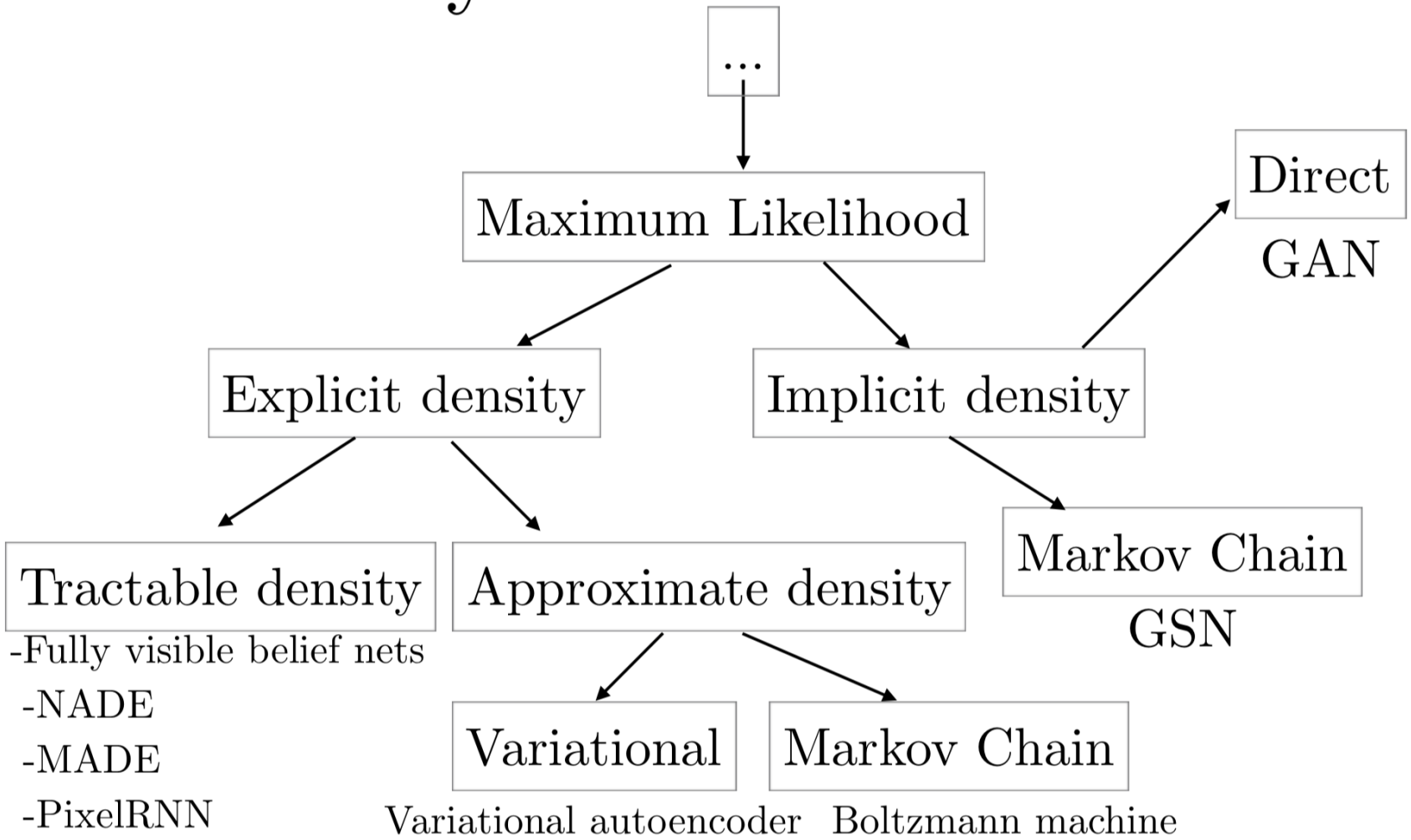
Training examples

Generated samples

Why study generative models?

- Excellent test of our ability to use high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
  - Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

Taxonomy of Generative Models

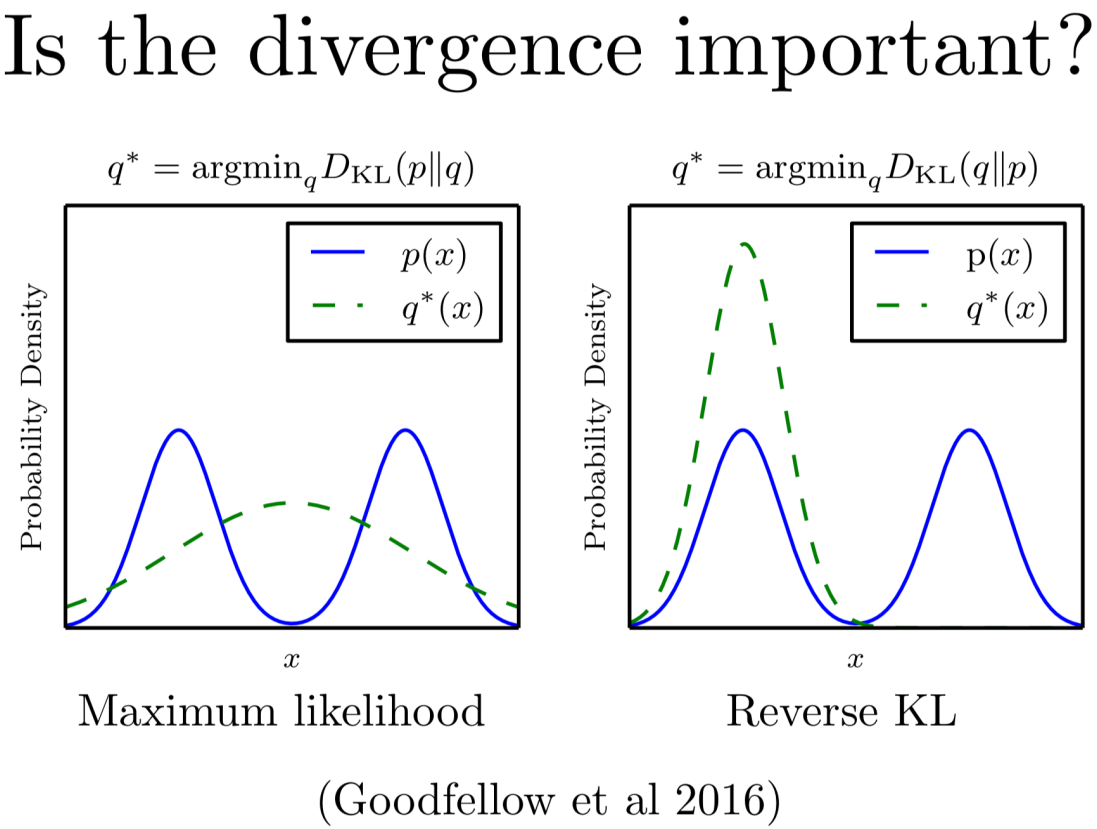


Limitations of VAEs

- Approximate density
- Set of family of distributions (the ones we can sample from)
- The KL loss

Limitations of VAEs

- VAEs tend to produce blurry images relative to GANs.



Jensen Shannon Divergence

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$$

Generative Matching Networks

- GMNs train the generative network by directly comparing the generated distribution to the true one **based on samples.**

- The MMD defines a distance between two probability distributions that can be computed (estimated) based on samples of these distributions.

How Computers mimic a random process ?

- Computers are fundamentally deterministic. So, it is, in theory, impossible to generate numbers that are really random (even if we could say that the question “what really is randomness ?” is a difficult one)

- However, it is possible to define algorithms that generate sequences of numbers whose properties are very close to the properties of theoretical random numbers sequences. In particular, a computer is able, using a pseudorandom number generator, to generate a sequence of numbers that approximatively follows a uniform random distribution between 0 and 1.

- The uniform case is a very simple one upon which more complex random variables can be built in different ways.

How Computers mimic a random process ?

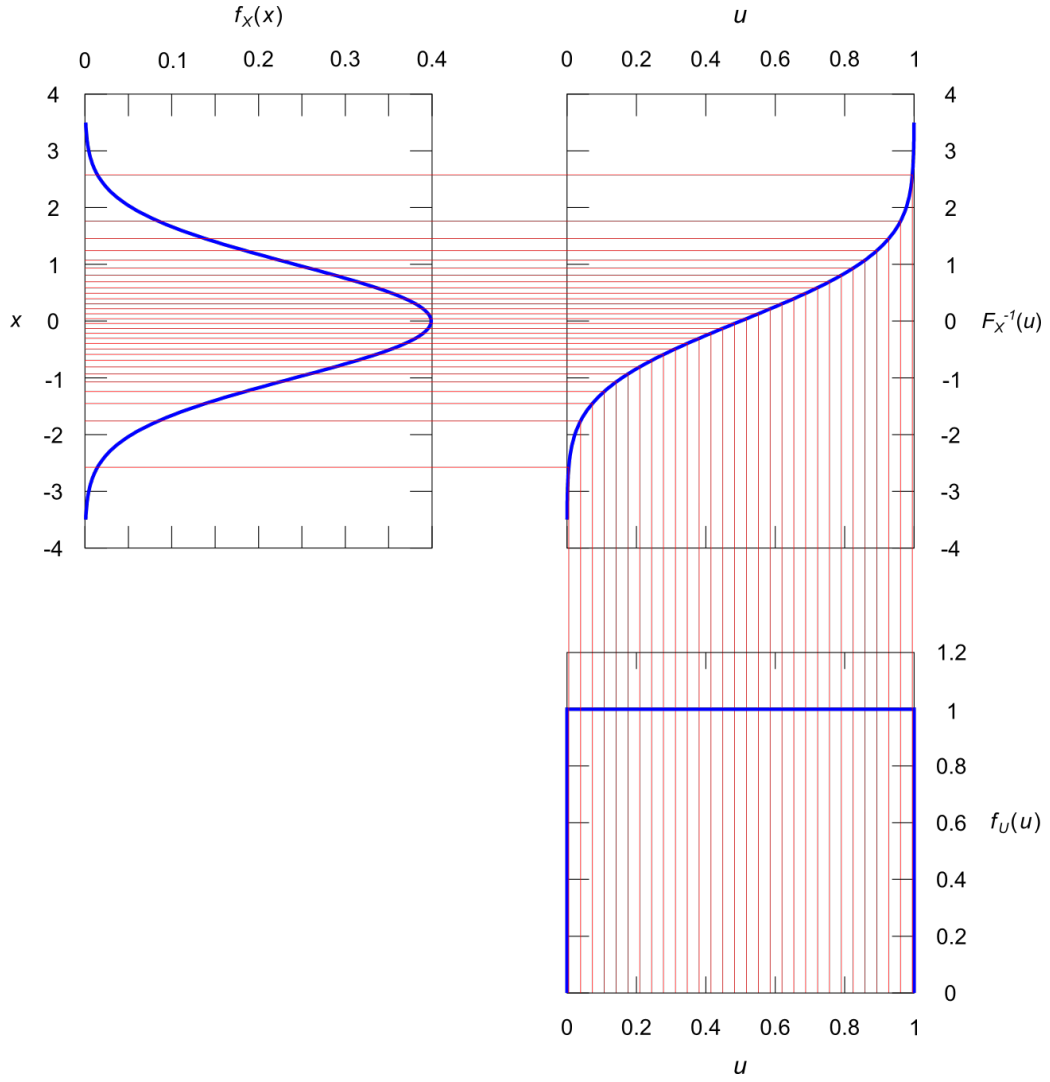
- How can I generate samples from normal distribution ?

- We can use inverse transform method, rejection sampling, Metropolis-Hasting algorithm and MCMC and other sampling methods.

The inverse transform method

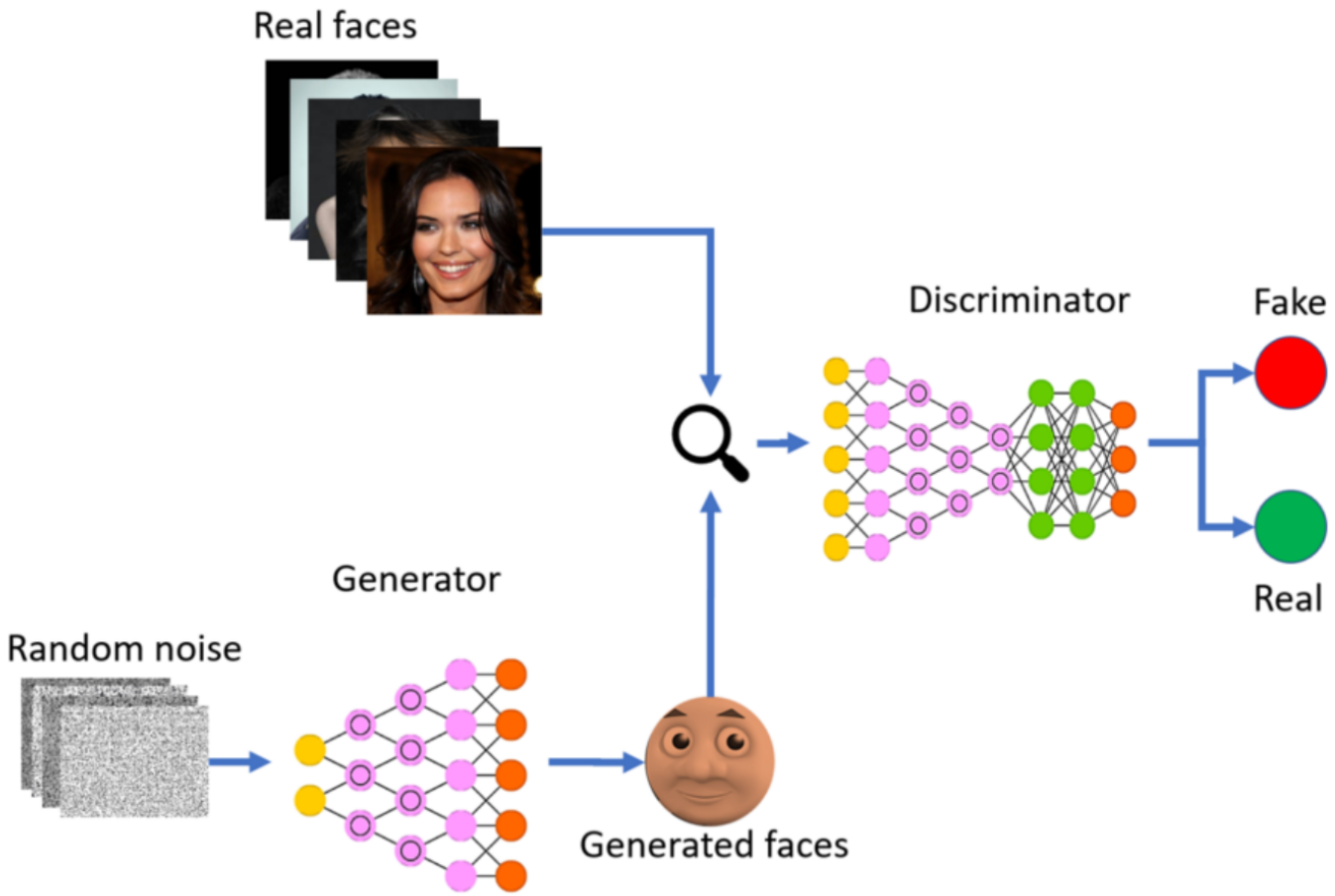
- We obtain a sample that follows a given distribution by sampling from a uniform random variable goes through a well designed “transform function”

- This notion of “inverse transform method” can, in fact, be extended to the notion of “transform method” that consists, more generally, in generating random variables as function of some simpler random variables (not necessarily uniform)



Adversarial training

- Frame the problem as a supervised learning problem



MinMax game

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$
$$J^{(G)} = -J^{(D)}$$

- You want the game to continue as much as it can
- Equilibrium is a saddle point of the discriminator loss

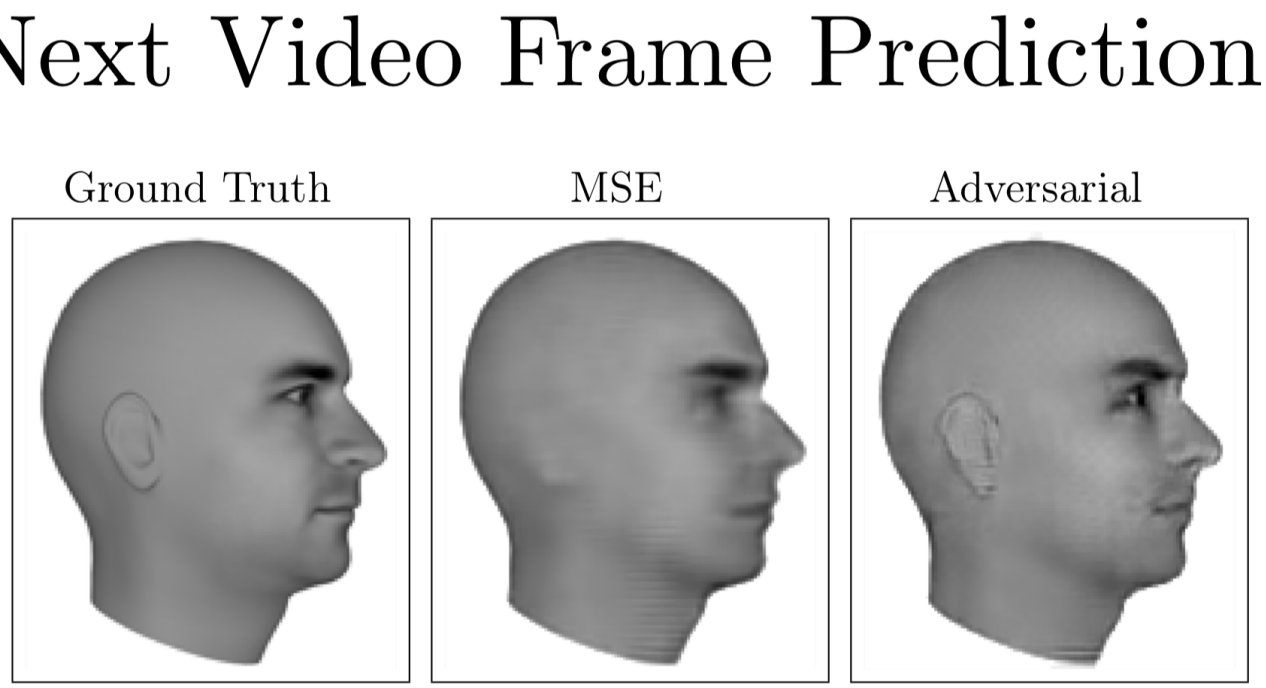
MinMax game

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -J^{(D)}$$

- Generator generate samples looks like they have been drawn from distribution of interest
- Discriminator is fooled 0.5 probability for all data points (fake or real)
- When discriminator is much better it will return only 0s or 1s and the generator will struggle to read the gradient.

MSE for generation



(Lotter et al 2016)

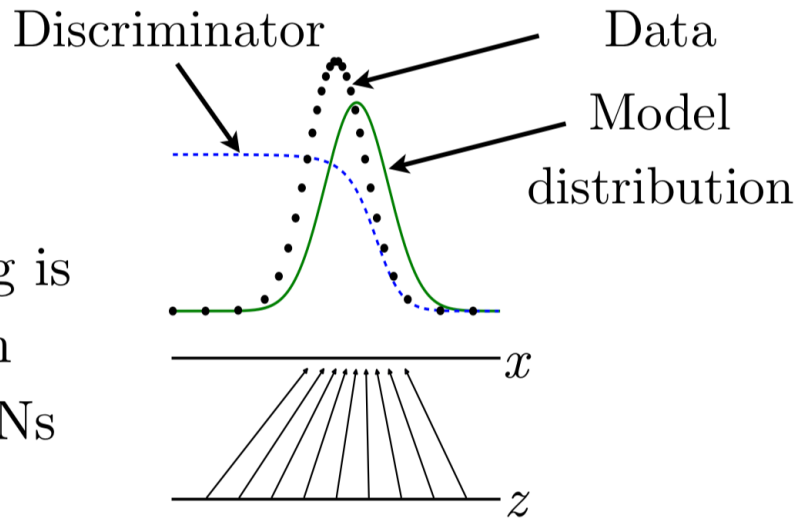
(Goodfellow 201(

Discriminator Strategy

Optimal $D(\boldsymbol{x})$ for any $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{model}}(\boldsymbol{x})$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Estimating this ratio using supervised learning is the key approximation mechanism used by GANs



Non-Saturating game

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

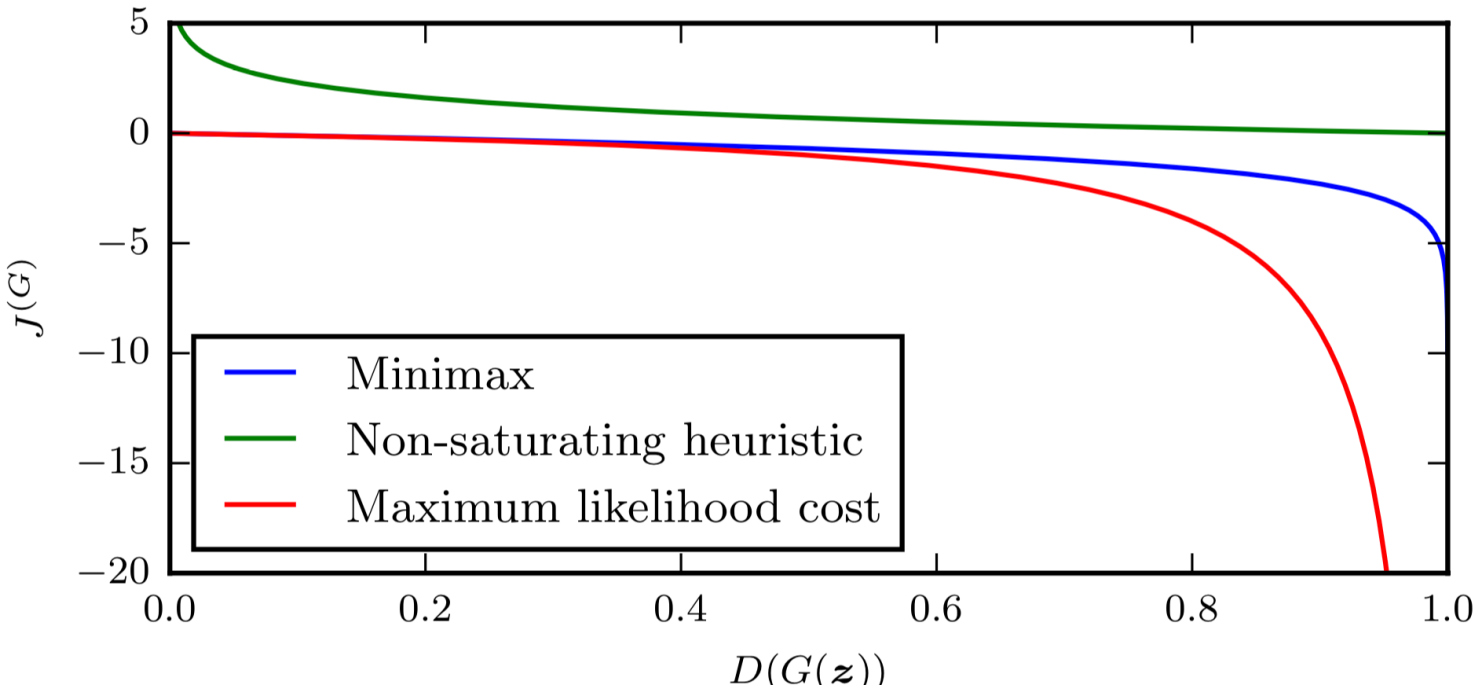
$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D\left(G(\boldsymbol{z})\right)$$

-Equilibrium no longer describable with a single loss

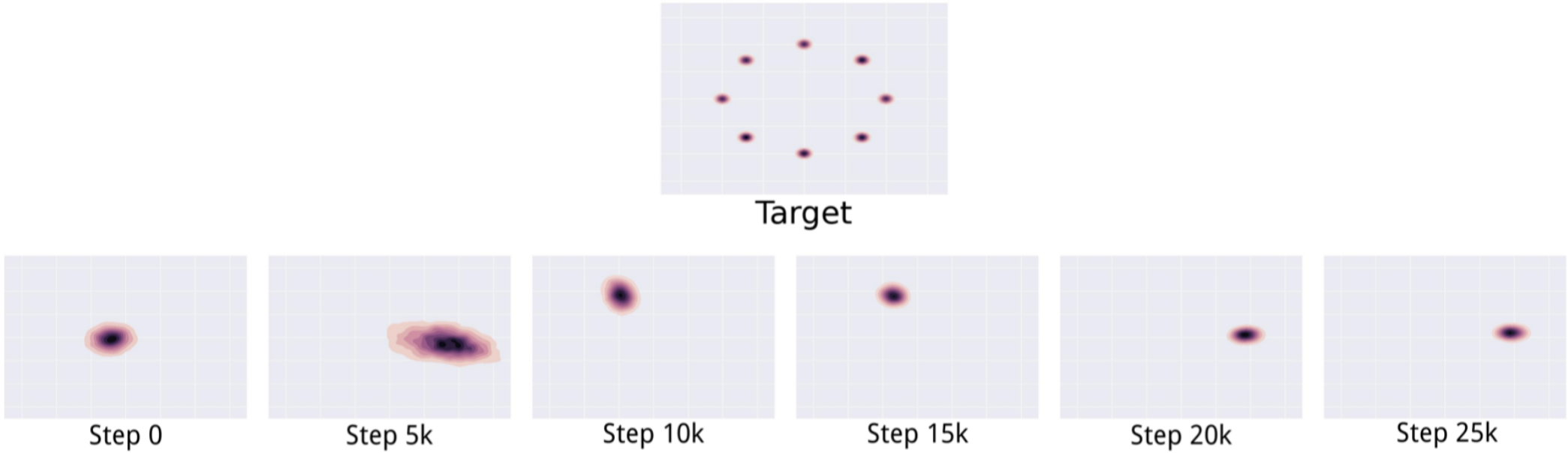
-Generator maximizes the log-probability of the discriminator being mistaken

-Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples

Compare Generator losses



Mode Collapse



- GANLAB

Metz et al 2016

Training GANs Tips and Tricks

- Usually the discriminator “wins”
- Usually *D* is bigger and deeper than *G*
- Sometimes run *D* more often than *G*. Mixed results.
- Do not try to limit *D* to avoid making it “too smart”

→ Use non-saturating cost

Training GANs Tips and Tricks

- Usually the discriminator “wins”
- Usually *D* is bigger and deeper than *G*
- Sometimes run *D* more often than *G*. Mixed results.
- Do not try to limit *D* to avoid making it “too smart”

→ Use non-saturating loss: generator can still learn even when discriminator successfully rejects all generator samples

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D\left(G(\boldsymbol{z})\right)$$

Labels improve subjective sample quality

- Learning a conditional model $p(y|x)$ often gives much better samples from all classes than learning $p(x)$ does (Denton et al 2015)
- Even just learning $p(x,y)$ makes samples from $p(x)$ look much better to a human observer (Salimans et al 2016)
- Note: this defines three categories of models (no labels, trained with labels, generating condition on labels) that should not be compared directly to each other

One-sided label

- Default discriminator cost:

```
    cross_entropy(1., discriminator(data))
+ cross_entropy(0., discriminator(samples))
```

- One-sided label smoothed cost (Salimans et al 2016):

```
    cross_entropy(.9, discriminator(data))
+ cross_entropy(0., discriminator(samples))
```

Don’t smooth on negative samples

```
cross_entropy(1.-alpha, discriminator(data))
+ cross_entropy(beta, discriminator(samples))
```

Reinforces current generator behavior

$$D(\boldsymbol{x}) = \frac{(1-\alpha)p_{\text{data}}(\boldsymbol{x}) + \beta p_{\text{model}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{model}}(\boldsymbol{x})}$$

Benefits of label smoothing

- Good regularizer (Szegedy et al 2015)
- Does not reduce classification accuracy, only confidence
- Benefits specific to GANs:
  - Prevents discriminator from giving very large gradient signal to generator
  - Prevents extrapolating to encourage extreme samples

Batch Norm

- Given inputs $X=\{x^{(1)}, x^{(2)}, .., x^{(m)}\}$
- Compute mean and standard deviation of features of X
- Normalize features (subtract mean, divide by standard deviation)
- Normalization operation is part of the graph
  - Backpropagation computes the gradient through the normalization
  - This avoids wasting time repeatedly learning to undo the normalization

Batch norm in G can cause strong intra-batch correlation

Reference Batch Norm

- Fix a *reference batch* $R=\{r^{(1)}, r^{(2)}, .., r^{(m)}\}$
- Given new inputs $X=\{x^{(1)}, x^{(2)}, .., x^{(m)}\}$
- Compute mean and standard deviation of features of R
  - Note that though R does not change, the feature values change when the parameters change
- Normalize the features of X using the mean and standard deviation from R
- Every $x^{(i)}$ is always treated the same, regardless of which other examples appear in the minibatch

Virtual Batch Norm

- Reference batch norm can overfit to the reference batch. A partial solution is *virtual batch norm*
- Fix a *reference batch* $R=\{r^{(1)}, r^{(2)}, .., r^{(m)}\}$
- Given new inputs $X=\{x^{(1)}, x^{(2)}, .., x^{(m)}\}$
- For each $x^{(i)}$ in X:
  - Construct a *virtual batch* V containing both $x^{(i)}$ and all of R
  - Compute mean and standard deviation of features of V
  - Normalize the features of $x^{(i)}$ using the mean and standard deviation from V

Minibatch features: addressing mode collapse

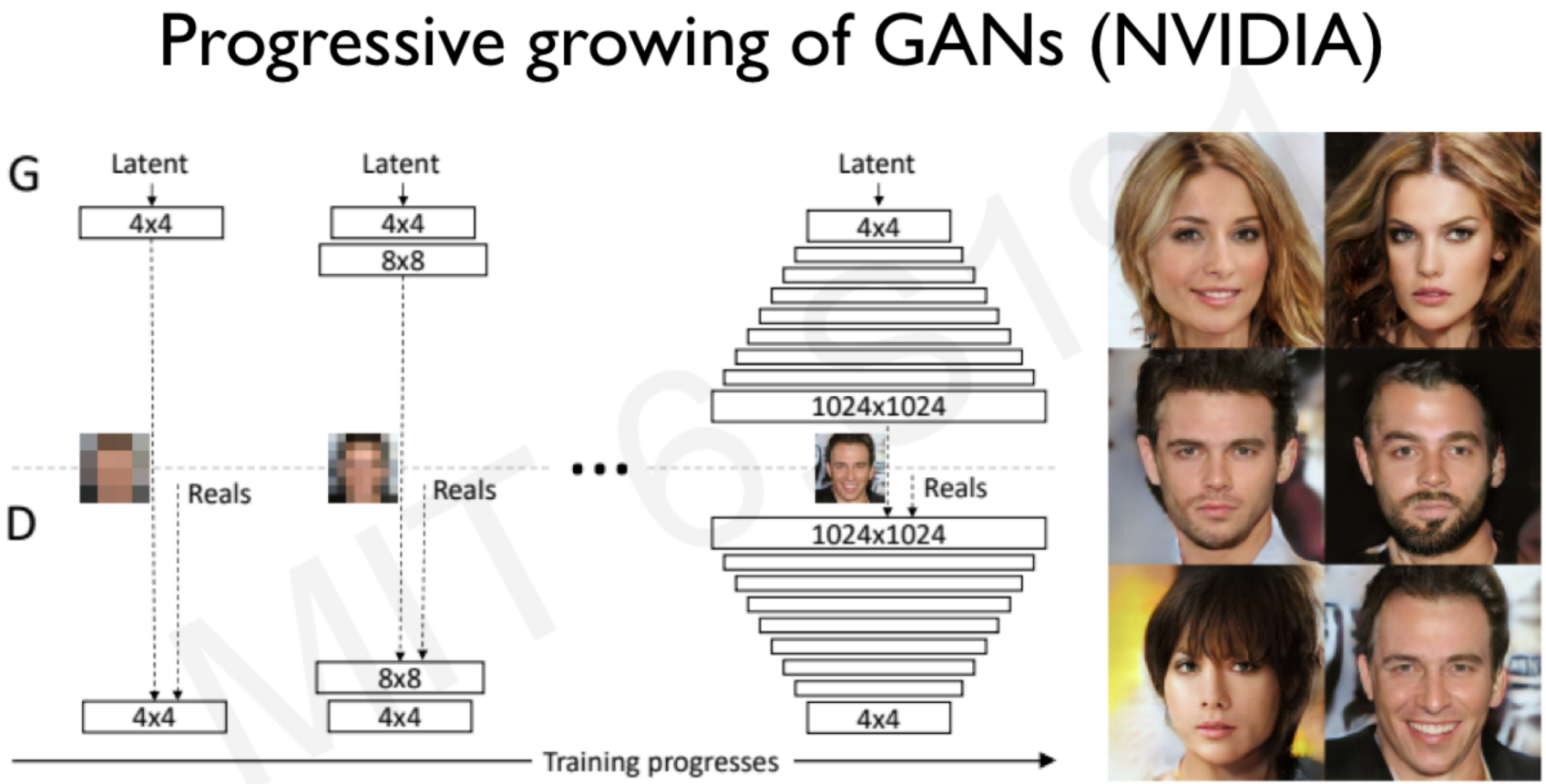
- Add minibatch features that classify each example by comparing it to other members of the minibatch (Salimans et al 2016)

- Nearest-neighbor style features detect if a minibatch contains samples that are too similar to each other

Popular GAN Papers

- DCGAN
- InfoGAN
- CycleGAN
- StyleGAN
- CGANs

Popular GAN architectures

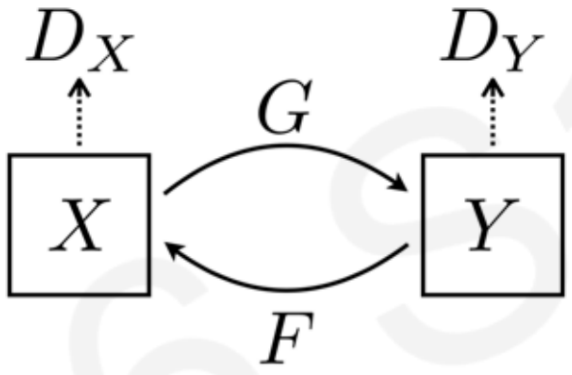


Popular GAN architectures

Popular GAN architectures

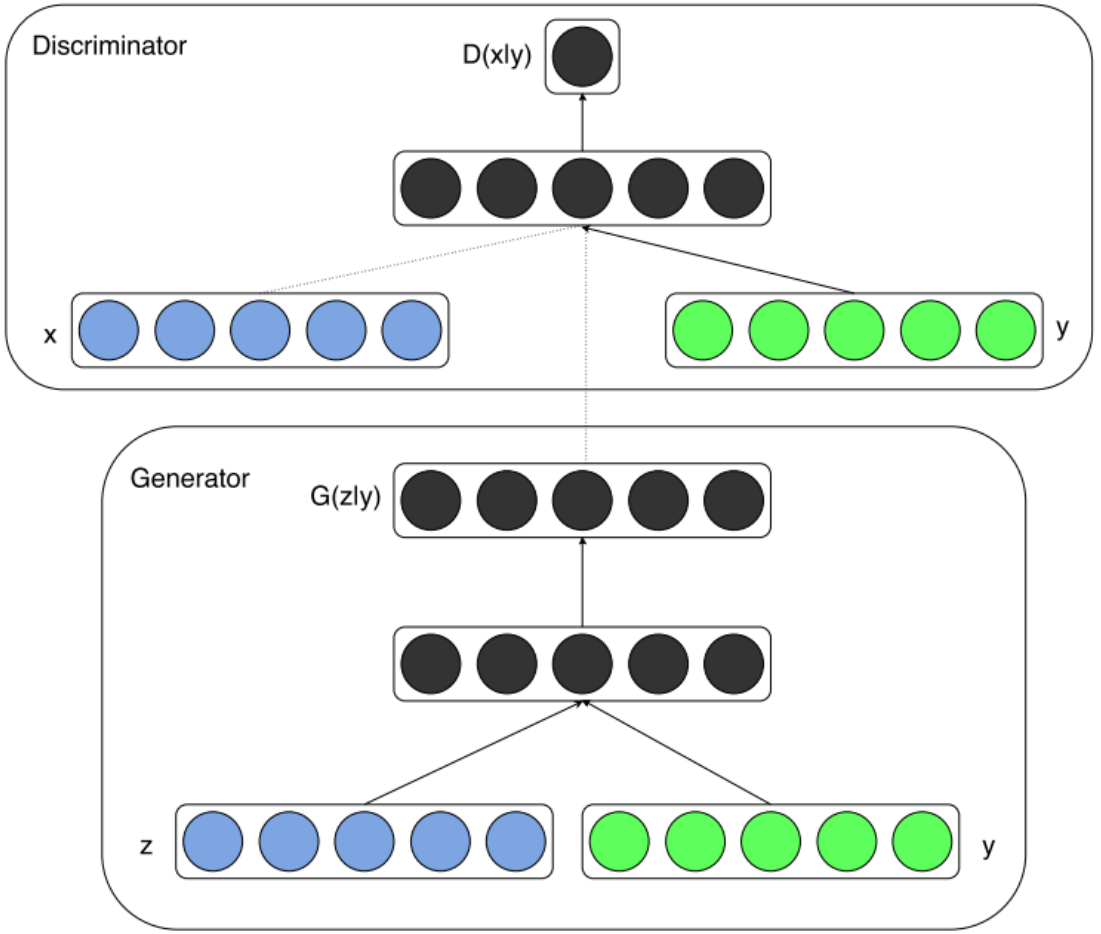
CycleGAN: domain transformation

CycleGAN learns transformations across domains with unpaired data.



Popular GAN architectures

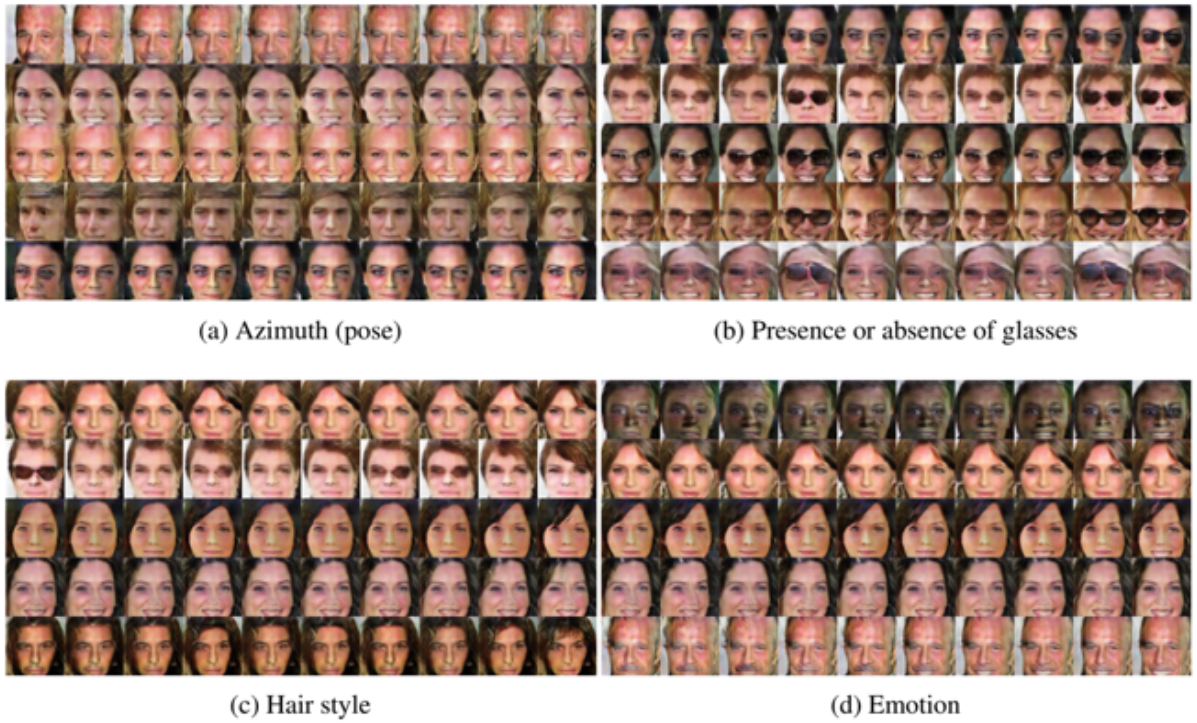
- CGAN



Popular GAN architectures

- Splits the Generator input into two parts: the traditional noise vector and a new “latent code” vector. The codes are then made meaningful by maximizing the Mutual Information between the code and the generator output.

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$



(a) Azimuth (pose)

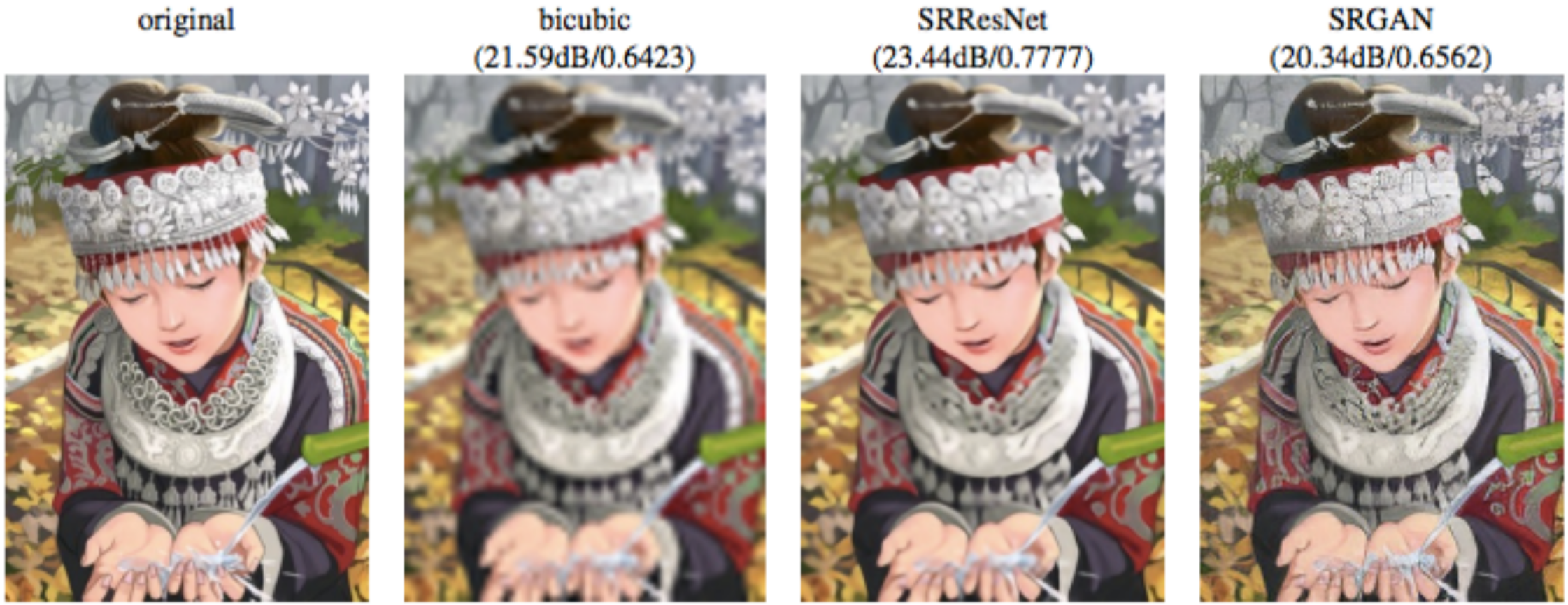
(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

Applications examples
Realistic samples generation

Applications examples
Super Resolution



Ledig et al 2016

Applications examples
Image to Image translation

Isola et al 2016

Applications examples
Style mix/transfer

References

- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville.*Deeplearning*.MITpress,2016.
- Ian Goodfellow: Generative Adversarial Networks (NIPS 2016 tutorial).
- Joseph Rocca, Understanding Generative Adversarial Networks (GANs).