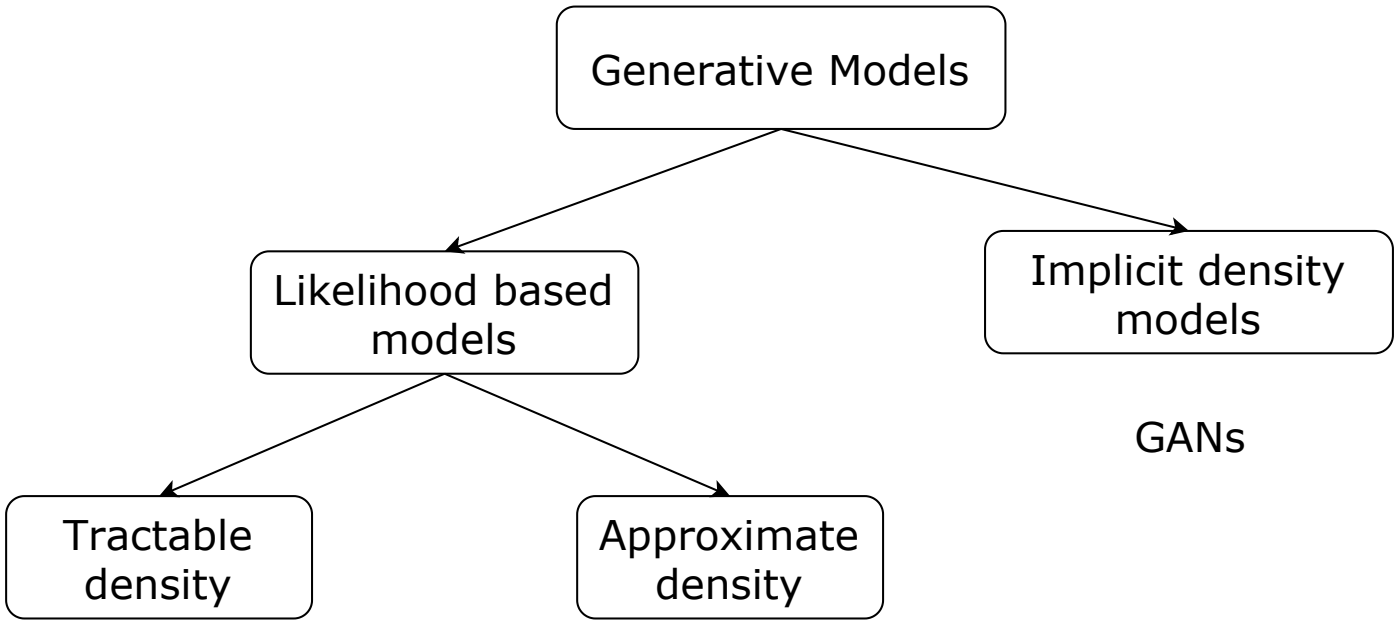
Deep Generative Models
Lecture 8

Roman Isachenko

Moscow Institute of Physics and Technology

2020

Generative models zoo



Likelihood based models

Is likelihood a good measure of model quality?

Poor likelihood
Great samples

$$p_1(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{N}(\mathbf{x}|\mathbf{x}_i, \epsilon\mathbf{I})$$

For small ϵ this model will generate samples with great quality, but likelihood will be very poor.

Great likelihood
Poor samples

$$p_2(\mathbf{x}) = 0.01p(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x})$$

$$\log\left[0.01p(\mathbf{x}) + 0.99p_{\text{noise}}(\mathbf{x})\right] \geq$$
$$\geq \log\left[0.01p(\mathbf{x})\right] = \log p(\mathbf{x}) - \log 100$$

Noisy irrelevant samples, but for high dimensions $\log p(\mathbf{x})$ becames larger.

---

https://arxiv.org/abs/1511.01844

Likelihood-free learning

- Likelihood is not a perfect measure for quality measure for generative model.
- Likelihood could be intractable.

Where did we start

We would like to approximate true data distribution $\pi(\mathbf{x})$. Instead of searching true $\pi(\mathbf{x})$ over all probability distributions, learn function approximation $p(\mathbf{x}|\boldsymbol{\theta}) \approx \pi(\mathbf{x})$.

Imagine we have two sets of samples

- $\mathcal{S}_1 = \{\mathbf{x}_i\}_{i=1}^{n_1} \sim \pi(\mathbf{x})$
- $\mathcal{S}_2 = \{\mathbf{x}_i\}_{i=1}^{n_2} \sim p(\mathbf{x}|\boldsymbol{\theta})$

Two sample test

$$H_0 : \pi(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}), \quad H_1 : \pi(\mathbf{x}) \neq p(\mathbf{x}|\boldsymbol{\theta})$$

Likelihood-free learning

Two sample test

$$H_0 : \pi(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}), \quad H_1 : \pi(\mathbf{x}) \neq p(\mathbf{x}|\boldsymbol{\theta})$$

Define test statistic $T(\mathcal{S}_1, \mathcal{S}_2)$. Test statictic is likelihood free.
If $T(\mathcal{S}_1, \mathcal{S}_2) < \alpha$, then accept H_0, else reject it.

Desired behaviour

- The generative model $p(\mathbf{x}|\boldsymbol{\theta})$ minimizes the value of test statistic $T(\mathcal{S}_1, \mathcal{S}_2)$.
- Find appropriate test statistic in high dimensions is hard. We could try to learn the appropriate $T(\mathcal{S}_1, \mathcal{S}_2)$.

Vanila GAN

- **Generator:** latent variable model $\mathbf{x} = G(\mathbf{z})$, which minimizes two-sample test objective.
- **Discriminator:** function $D(\mathbf{x})$, which distinguish real samples from model samples and maximizes two-sample test statistic.

Objective

$$V(G, D) = \min_G \max_D \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

For fixed generator $G(\mathbf{z})$ discriminator is performing binary classification with cross entropy loss.
This minimax game has global optimum $\pi(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta})$.

---

https://arxiv.org/abs/1406.2661

Vanila GAN optimality

Objective (fixed G)

$$\max_D V(D) = \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

Optimal discriminator

$$\begin{aligned} V(G, D) &= \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})} \log(1 - D(\mathbf{x})) \\ &= \int \underbrace{[\pi(\mathbf{x}) \log D(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta}) \log(1 - D(\mathbf{x})]}_{y(D)} \, d\mathbf{x} \end{aligned}$$

$$\frac{dy(D)}{dD} = \frac{\pi(\mathbf{x})}{D(\mathbf{x})} - \frac{p(\mathbf{x}|\boldsymbol{\theta})}{1 - D(\mathbf{x})} = 0 \quad \Rightarrow \quad D^*(\mathbf{x}) = \frac{\pi(\mathbf{x})}{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}$$

---

https://arxiv.org/abs/1406.2661

Vanila GAN optimality

Objective (fixed D)

$$\max_G V(G, D^*) = \mathbb{E}_{\pi(\mathbf{x})} \log D^*(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D^*(G(\mathbf{z})))$$

Optimal generator

$$V(G, D^*) = \mathbb{E}_{\pi(\mathbf{x})} \log \frac{\pi(\mathbf{x})}{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})} + \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})} \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}$$

$$= KL\left(\pi(x)||\frac{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}{2}\right) + KL\left(p(x|\boldsymbol{\theta})||\frac{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}{2}\right) - 2\log 2$$

$$= 2JSD(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) - 2\log 2.$$

---

https://arxiv.org/abs/1406.2661

Vanila GAN optimality

Optimal generator

$$V(G, D^*) = 2JSD(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) - 2\log 2.$$

Jensen-Shannon divergence

$$JSD(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) = \frac{1}{2}\left[KL\left(\pi(x)||\frac{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}{2}\right) + KL\left(p(x|\boldsymbol{\theta})||\frac{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}{2}\right)\right]$$

Also called symmetric KL divergence. Could be used as a distance measure!

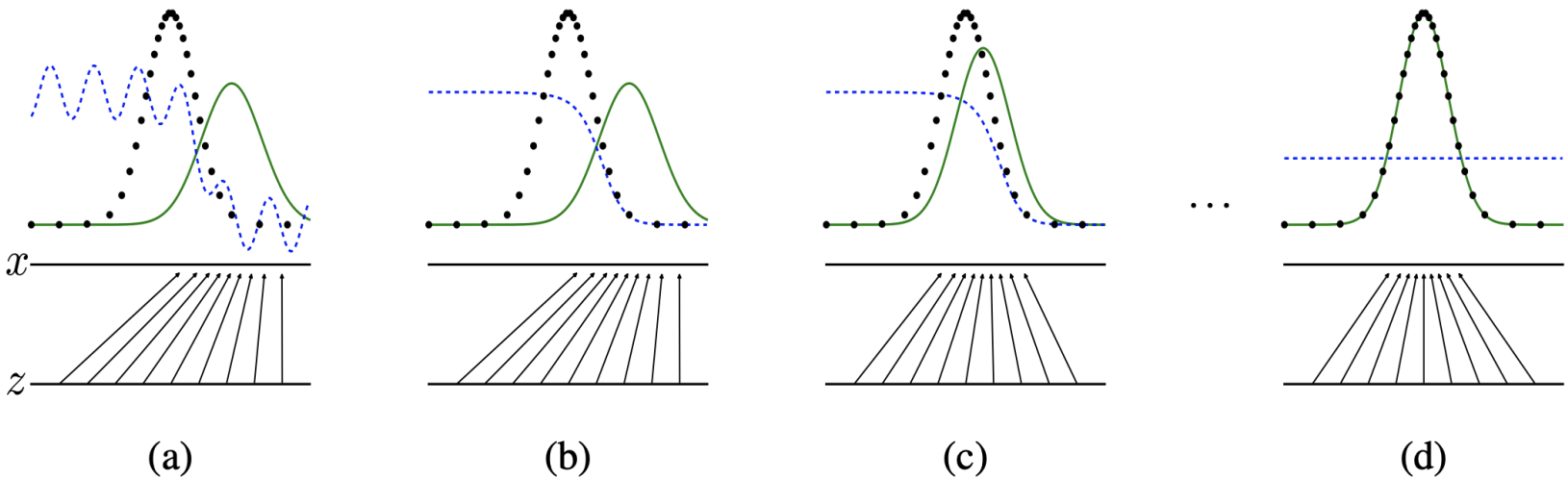
$$V(G^*, D^*) = -2\log 2, \quad \pi(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}).$$

If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution.

Vanila GAN

Objective

$$V(G, D) = \min_G \max_D \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$



- Generator updates are made in parameter space.
- Discriminator is not optimal at every step.
- Generator and discriminator loss keeps oscillatingduring GAN training.

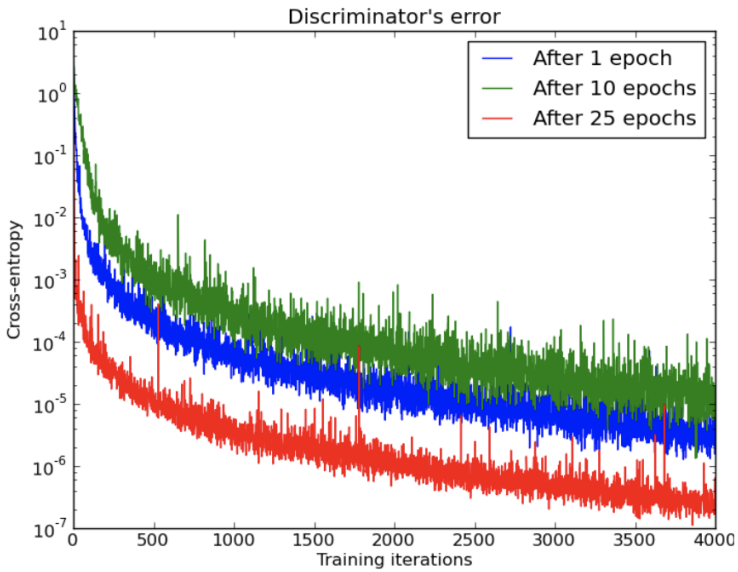
https://arxiv.org/abs/1406.2661

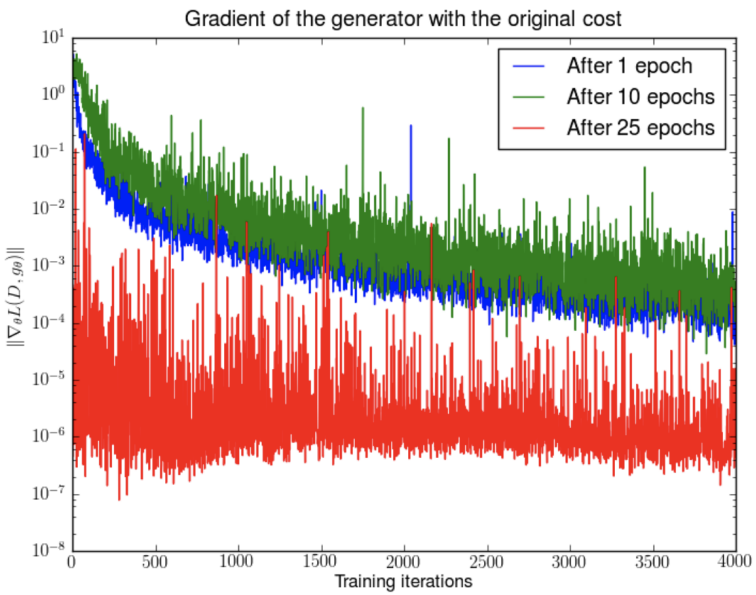
Vanishing gradients

Objective

$$V(G, D) = \min_G \max_D \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

Early in learning, G is poor, D can reject samples with high confidence. In this case, $\log(1 - D(G(\mathbf{z})))$ saturates.





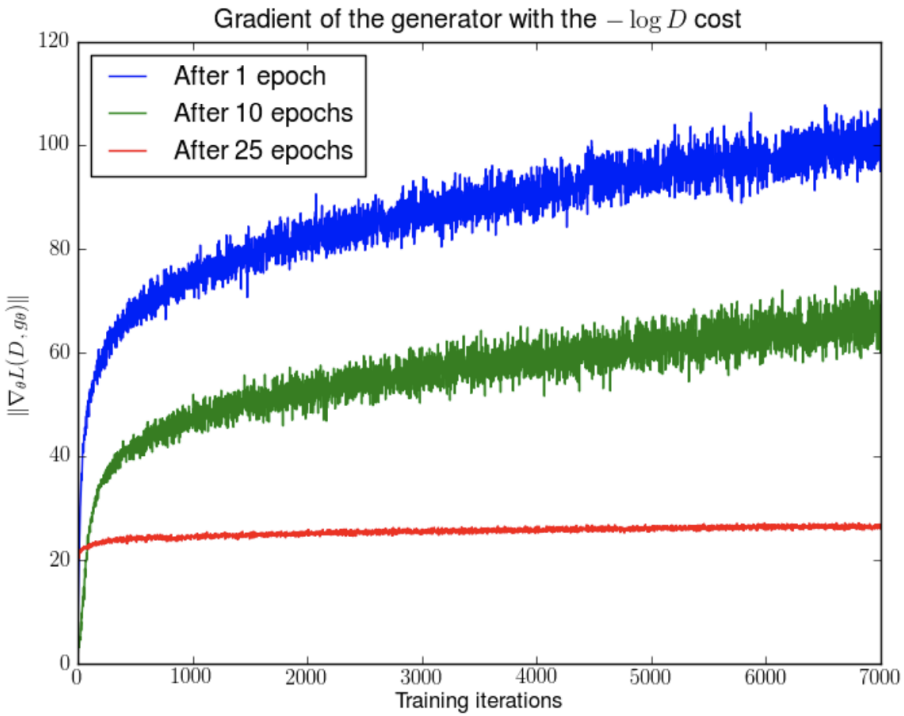
https://arxiv.org/abs/1701.04862

Vanishing gradients

Objective

$$V(G, D) = \min_G \max_D \mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

Maximize $\log D(G(z))$ instead of $\log(1 - D(G(\mathbf{z})))$.
Gradients are getting much stronger, but the training is instable (with increasing mean and variance).



https://arxiv.org/abs/1701.04862

Likelihood-based vs GAN

GAN objective (for optimal discriminator)

$$V(G, D^*) = 2JSD(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) - \log 4.$$

Likelihood model objective

$$\max_{\boldsymbol{\theta}} \log p(\mathbf{X}|\boldsymbol{\theta}) \approx \max_{\boldsymbol{\theta}} \mathbb{E}_{\pi(\mathbf{x})} \log p(\mathbf{x}|\boldsymbol{\theta}) =$$

$$= \max_{\boldsymbol{\theta}} \mathbb{E}_{\pi(\mathbf{x})} \log p(\mathbf{x}|\boldsymbol{\theta}) + \mathbb{E}_{\pi(\mathbf{x})} \log \pi(\mathbf{x}) =$$

$$= \max_{\boldsymbol{\theta}} \mathbb{E}_{\pi(\mathbf{x})} \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x})} = \min_{\boldsymbol{\theta}} KL(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta}))$$

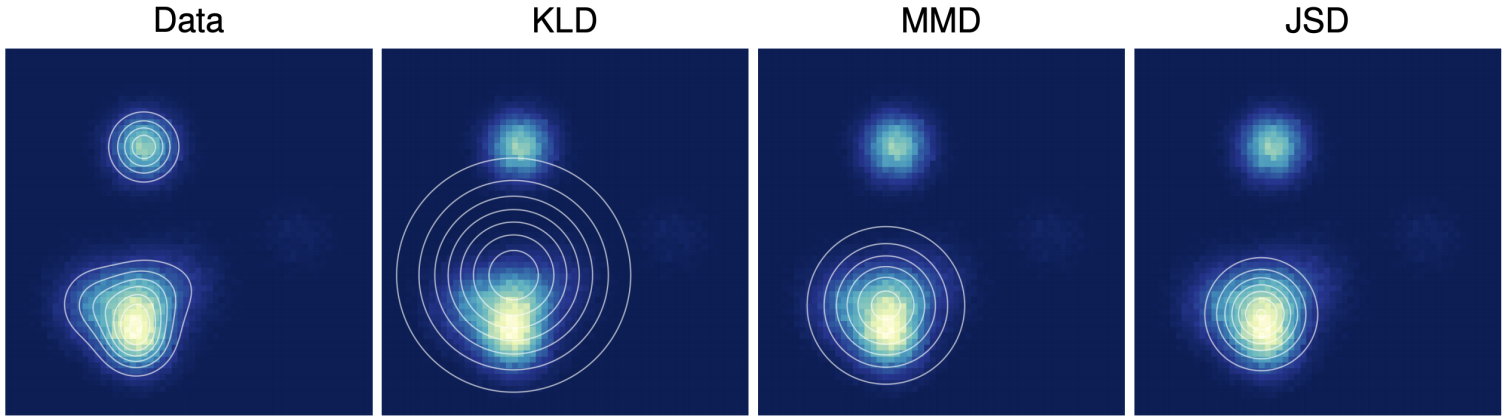
What is the difference between JS and KL divergences?

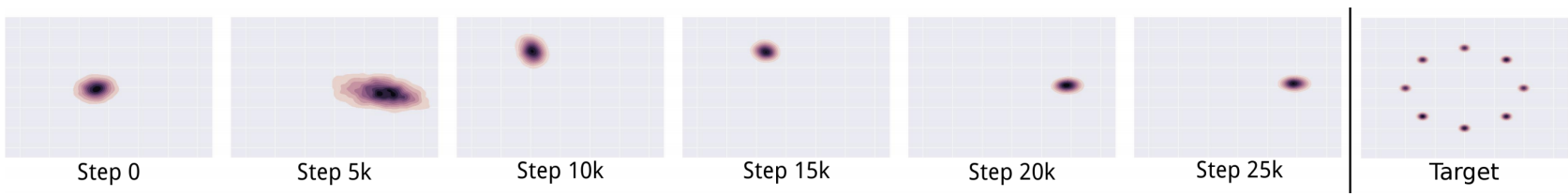
---

https://arxiv.org/abs/1511.01844

Mode collapse

The phenomena where the generator of a GAN collapses to one or few distribution modes.





Alternate architectures, adding regularization terms, injecting small noise perturbations and other millions bags and tricks are used to avoid the mode collapse.

https://arxiv.org/abs/1406.2661

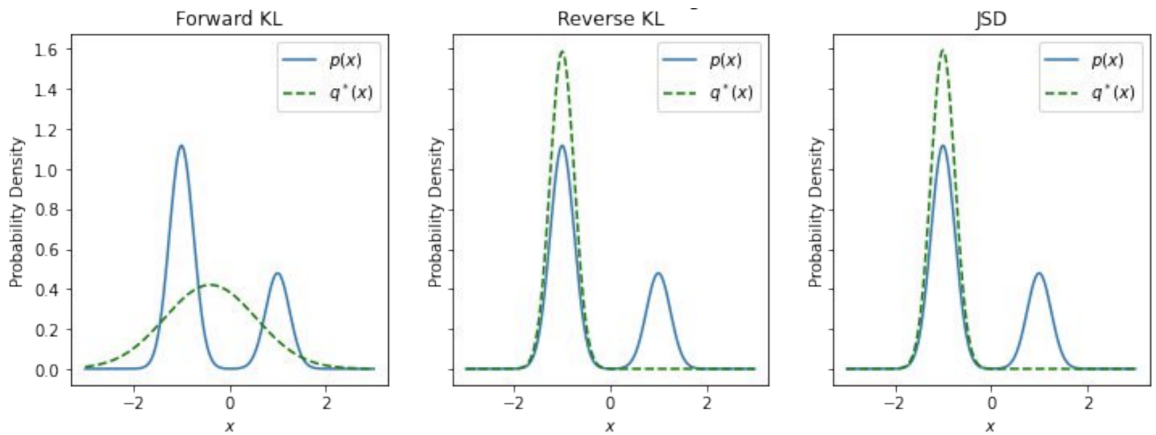
https://arxiv.org/abs/1611.02163

Mode collapse

Mode covering vs mode seeking

$$KL(\pi||p) = \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x}, \quad KL_r(p||\pi) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{\pi(\mathbf{x})} d\mathbf{x}$$

$$JSD(\pi||p) = \frac{1}{2}\left[KL\left(\pi(x)||\frac{\pi(\mathbf{x}) + p(\mathbf{x})}{2}\right) + KL\left(p(x)||\frac{\pi(\mathbf{x}) + p(\mathbf{x})}{2}\right)\right]$$

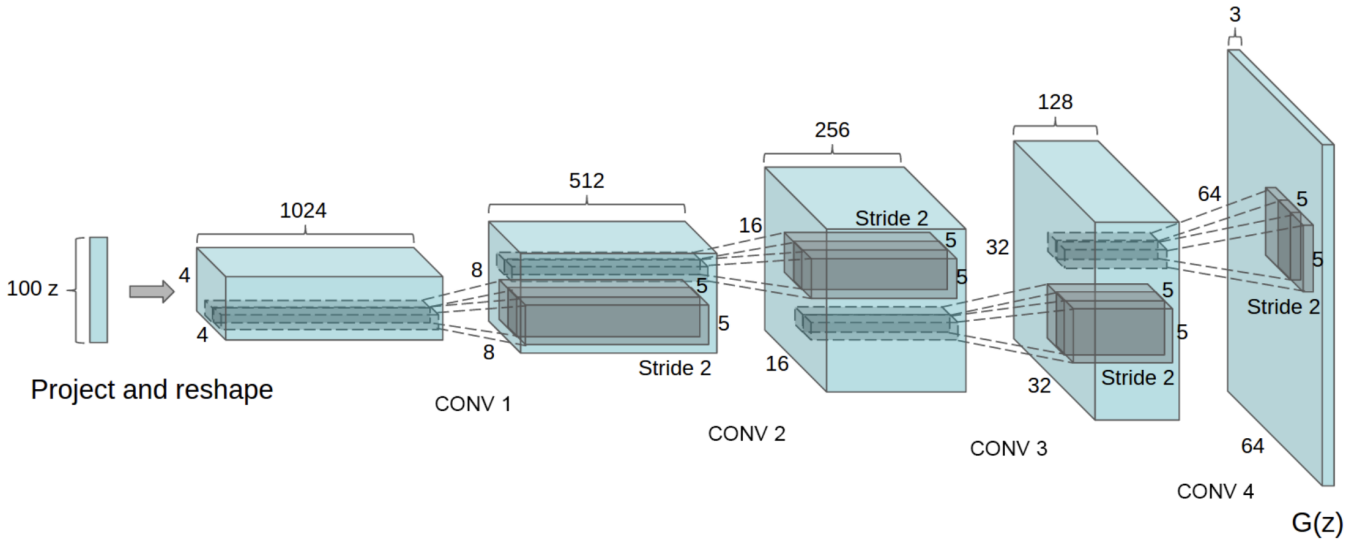


https://sites.google.com/view/berkeley-cs294-158-sp20/home

Vanila GAN results

https://arxiv.org/abs/1406.2661

Deep Convolutional GAN



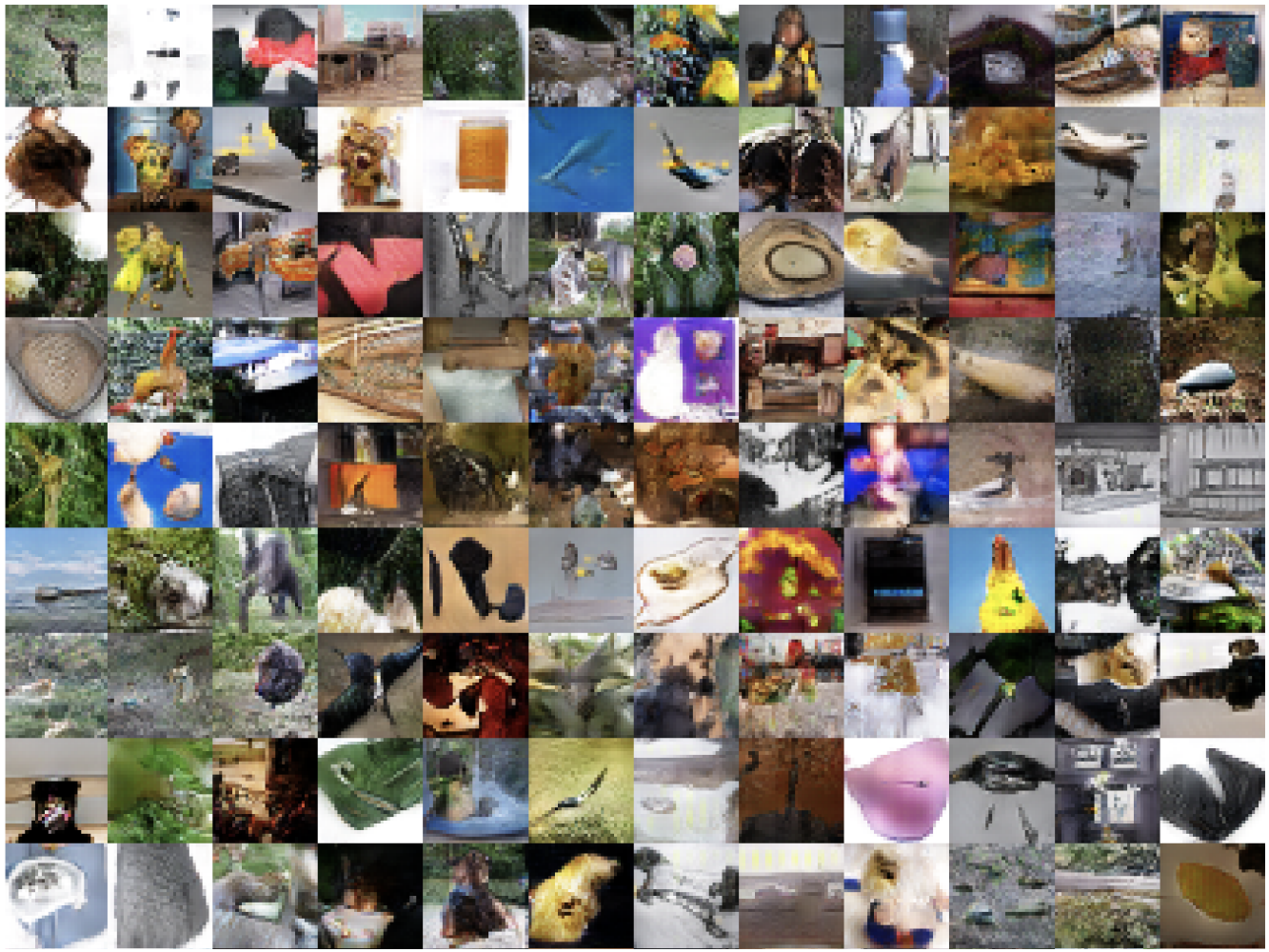
- Mean-pooling instead of max-pooling.
- Transposed convolutions in the generator for upsampling.
- Downsample with strided convolutions and average pooling.
- ReLU for generator, Leaky-ReLU (0.2) for discriminator.
- Output nonlinearity: tanh for Generator, sigmoid for discriminator.
- Batch Normalization used to prevent mode collapse (not applied at the output of G and input of D).
- Adam: small LR = 2e-4; small momentum: 0.5, batch-size: 128.

---

https://arxiv.org/abs/1511.06434

Deep Convolutional GAN

ImageNet samples



https://arxiv.org/abs/1511.06434

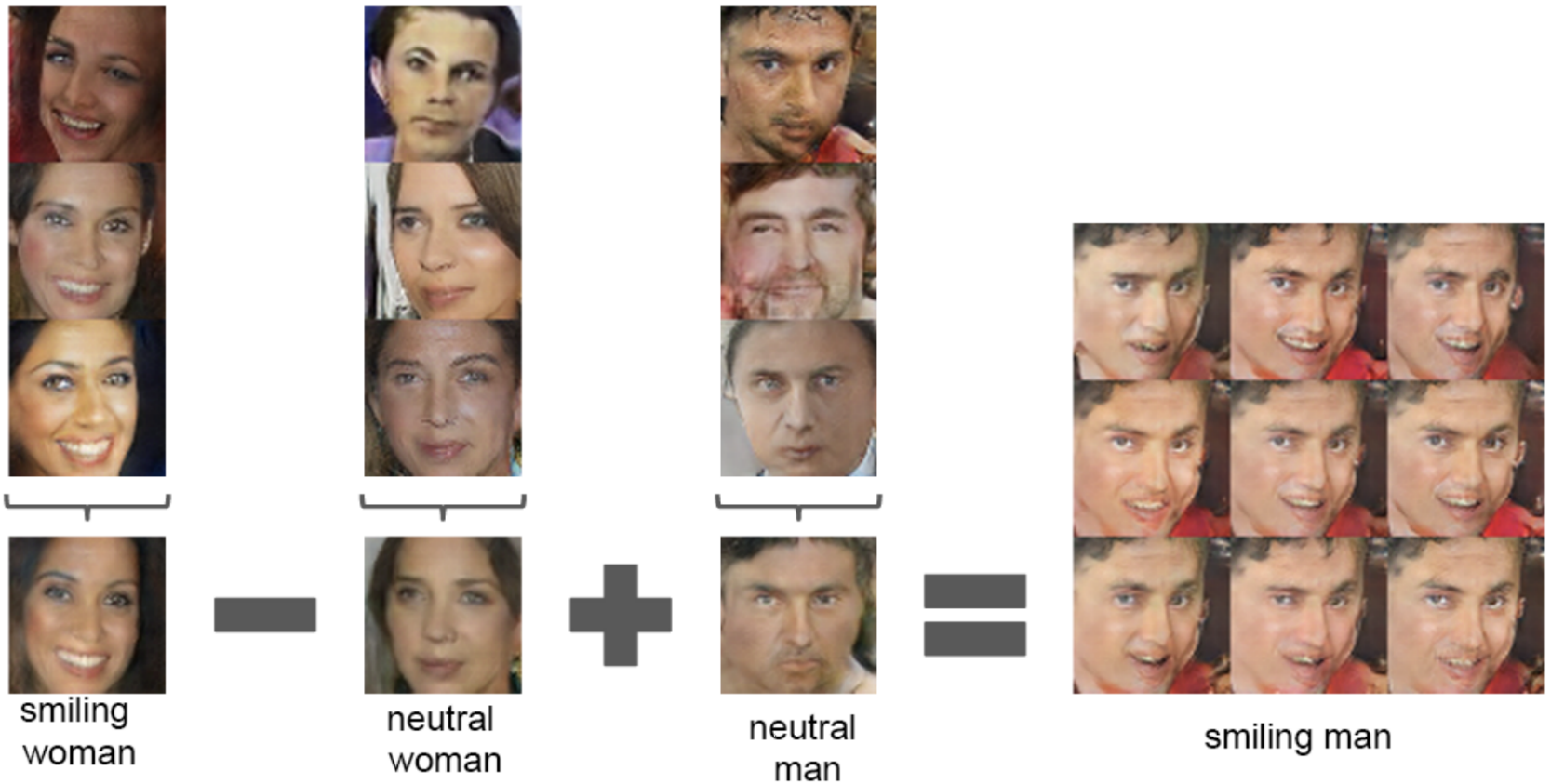
Deep Convolutional GAN

Smooth interpolations

https://arxiv.org/abs/1511.06434

Deep Convolutional GAN

Vector arithmetic



https://arxiv.org/abs/1511.06434

Improved techniques for training GANs

- Feature matching

$$\mathcal{L}_G = \|\mathbb{E}_{\pi(\mathbf{x})}\mathbf{d}(\mathbf{x}) - \mathbb{E}_{p(\mathbf{z})}\mathbf{d}(G(\mathbf{z}))\|_2^2$$

  Here $\mathbf{d}(\mathbf{x})$ – intermediate layer of discriminator. Matching the learned discriminator statistics instead of the output of the discriminator. Helps to avoid the vanishing gradients for sufficiently good discriminator.

- Historical averaging adds extra loss term for generator and discriminator losses

$$\|\boldsymbol{\theta} - \sum_{t=1}^{T} \boldsymbol{\theta}_t\|_2^2.$$

  Here $\boldsymbol{\theta}_t$ – value of parameters at the previous step t. It allows to stabilize training procedure.

---

https://arxiv.org/abs/1606.03498

Improved techniques for training GANs

- One-sided label smoothing. Instead of using one-hot labels in classification, use $(1 - \alpha)$ for real data (the generated samples are not smoothed).

$$D^*(\mathbf{x}) = \frac{(1-\alpha)\pi(\mathbf{x})}{\pi(\mathbf{x}) + p(\mathbf{x}|\boldsymbol{\theta})}$$

- Virtual batch normalization. BatchNorm makes samples within minibatch are highly correlated.

Use reference fixed batch to compute the normalization statistics. To avoid overfitting construct batch with the reference batch and the current sample.

---

https://arxiv.org/abs/1606.03498

References

- A note on the evaluation of generative models
  https://arxiv.org/abs/1511.01844
  Summary:
- Generative Adversarial Nets
  https://arxiv.org/abs/1406.2661
  Summary:
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
  https://arxiv.org/abs/1511.06434
  Summary:
- Improved techniques for training GANs
  https://arxiv.org/abs/1606.03498
  Summary:
- Towards principled methods for training generative adversarial networks
  https://arxiv.org/abs/1701.04862
  Summary: