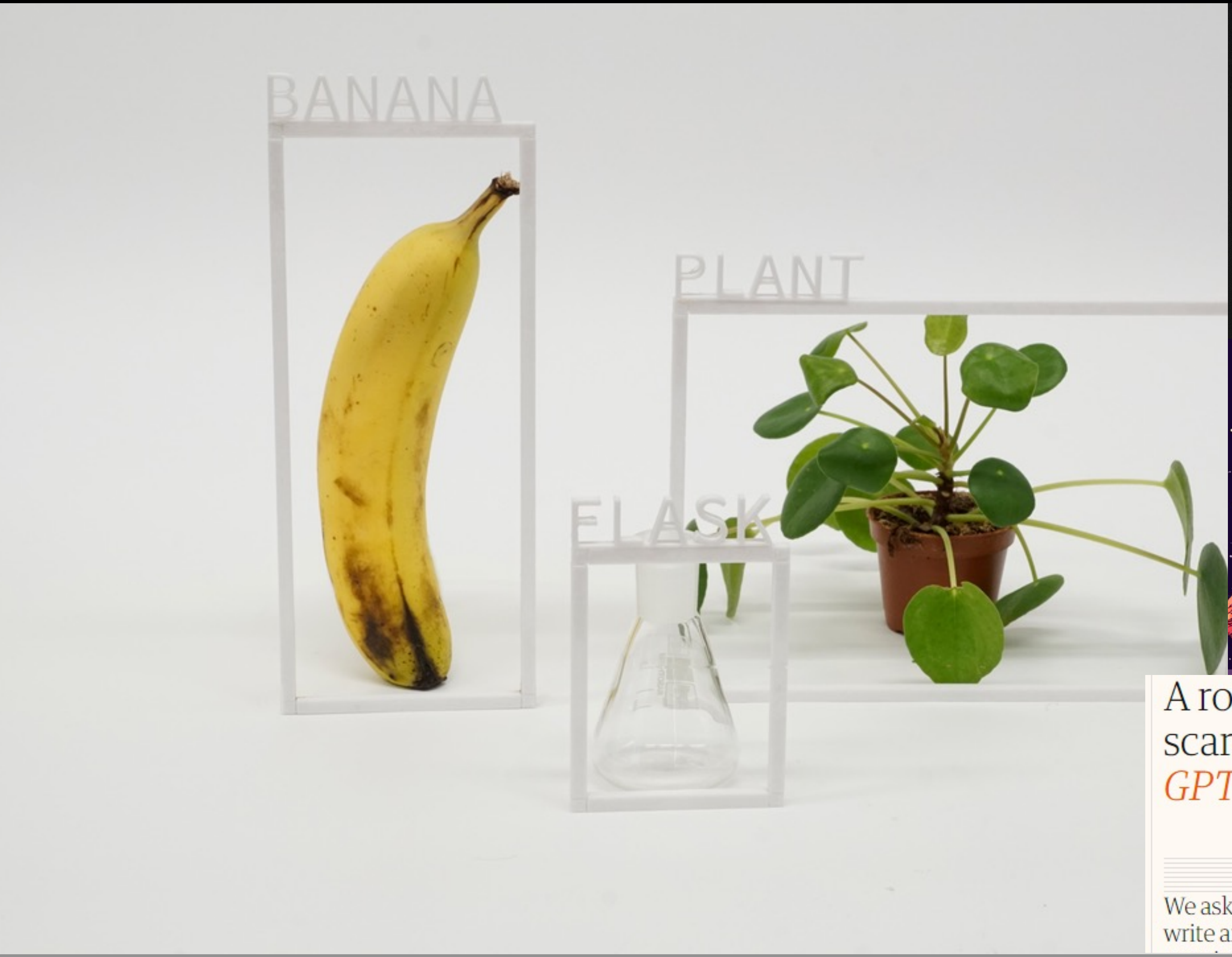
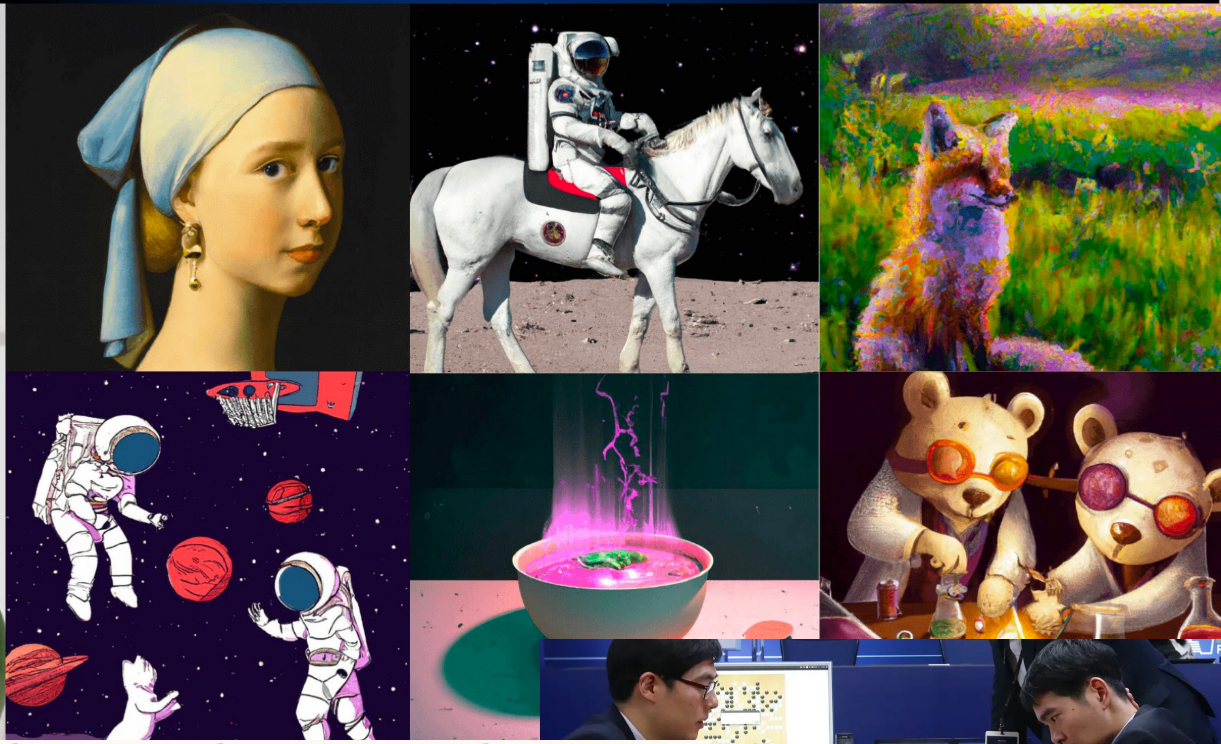
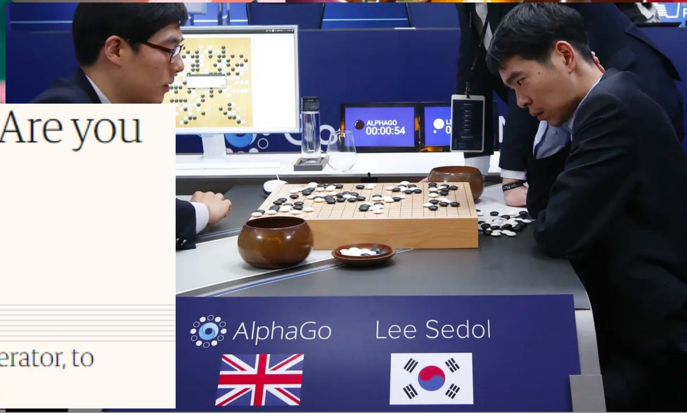




A robot wrote this entire article. Are you scared yet, human?

GPT-3

We asked GPT-3, OpenAI's powerful new language generator, to write an essay for us from scratch. The assignment? To



Lecture 10: Generative Adversarial Networks and Diffusion models

Deep Learning 1 @ UvA
Yuki M. Asano

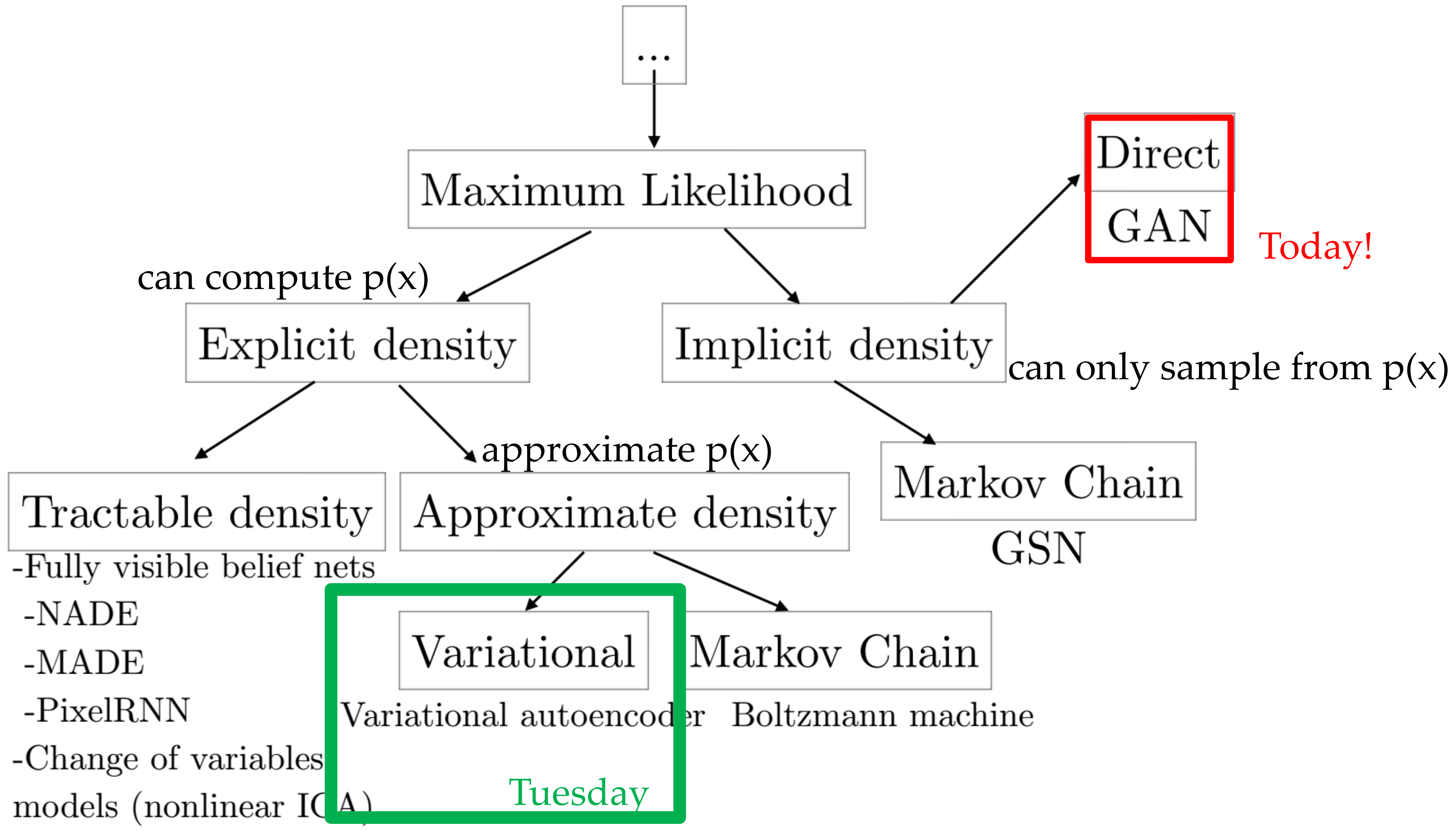
Organisation

- Guest Lecture on 6th December will be remote: https://uva-live.zoom.us/j/6466222109
- Please be there in-person for lecture on the 13th December
- Assignment 3 will be released today; deadline: 13th December 23:59

Lecture overview

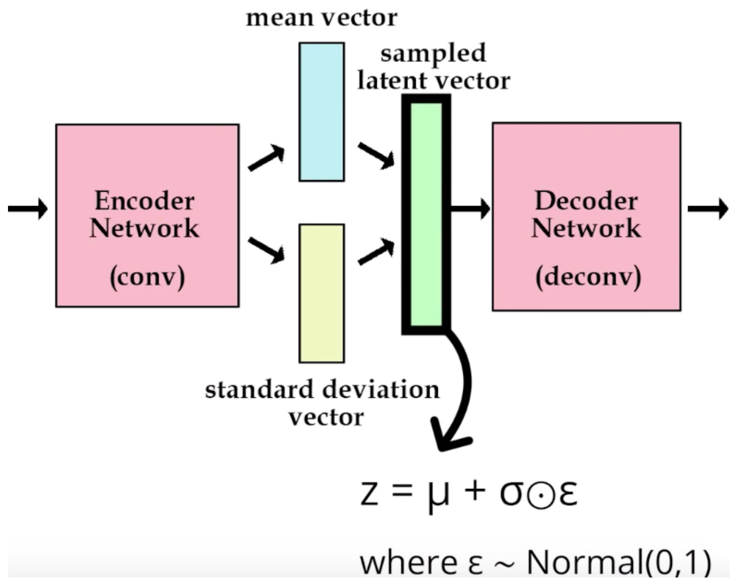
- Implicit density models: Motivation
- Generative adversarial networks
- Challenges
- GAN models
- Primer on diffusion models

A map of generative models



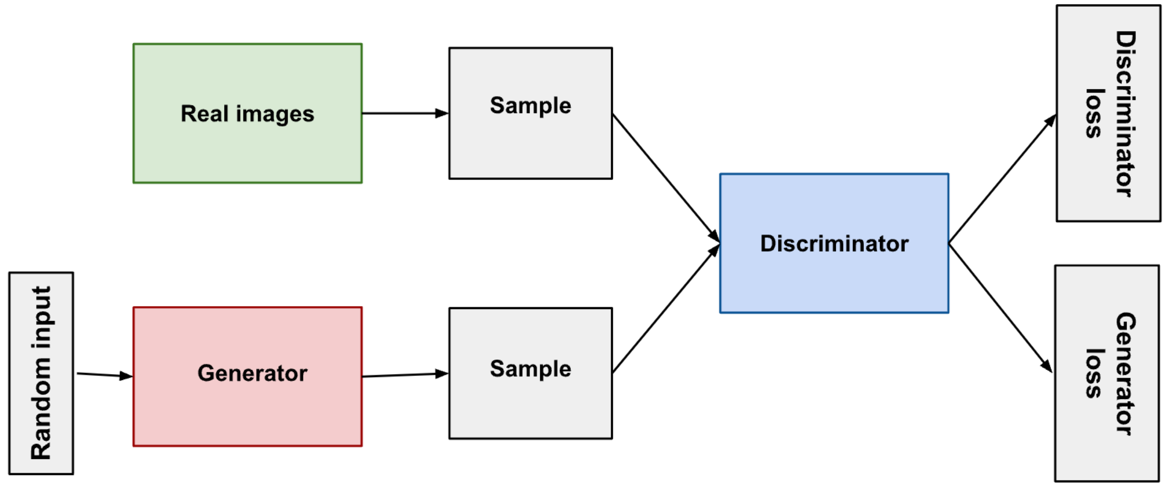
Last time

- All of the ELBO, amortization, reparametrization for:
  - Sampling from what-is-almost-an autoencoder (Encoder-Decoder)
  - Arriving at a (probabilistic) explicit density generative model



Today

- ~~All of the ELBO, amortization, reparametrization for:~~
  - ~~Sampling from what-is-almost-an autoencoder (Encoder-Decoder)~~
    - Well, now we will have something that will look like Decoder-Encoder
  - ~~Arriving at a (probabilistic) explicit density generative model~~
- But we'll still be able to sample from it



Explicit density vs implicit density

With $p^*(x)$ being the real distribution:

- The model p_θ assigns high density to samples taken from the true distribution p^*:

$$\boldsymbol{x} \sim p^*(\boldsymbol{x}) \implies p_\theta(\boldsymbol{x}) \text{ is "high".}$$

Explicit density

- Samples taken from the model p_θ behave similarly to real samples from p^*:

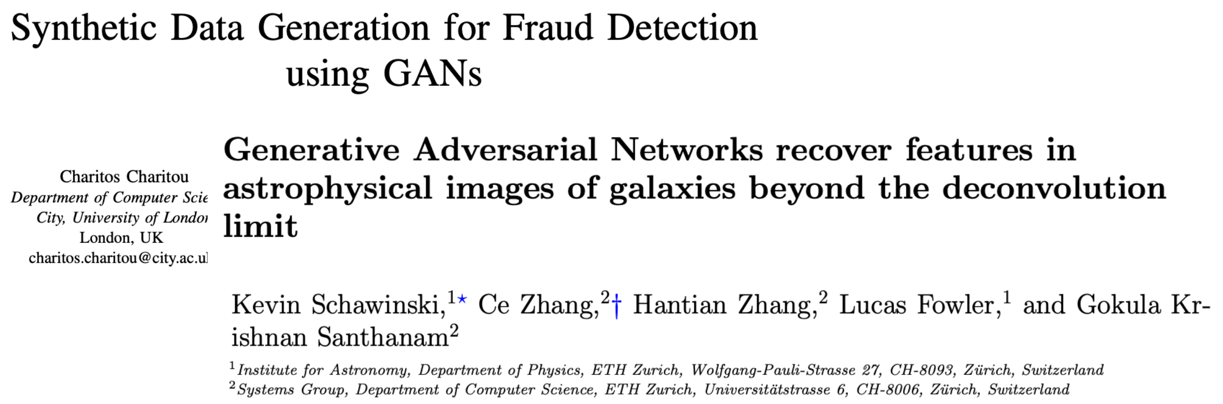
$$\boldsymbol{x} \sim p_\theta(\boldsymbol{x}) \implies p^*(\boldsymbol{x}) \text{ is "high".}$$

Implicit density

Learning an implicit density function

- Generally, learning this $p_\theta(x)$ is hard (you've survived Tuesday!)
- Instead, learn evaluate directly if the generations are plausible
  - And return gradients when not
- What is a plausible generation?
  - Especially in an unsupervised setting with no guidance
- Idea: use another "adversarial" network that tries to distinguish between generated and real data.
- Combined, these are called Generative Adversarial Networks (GANs)

Generations of high quality, various potential applications



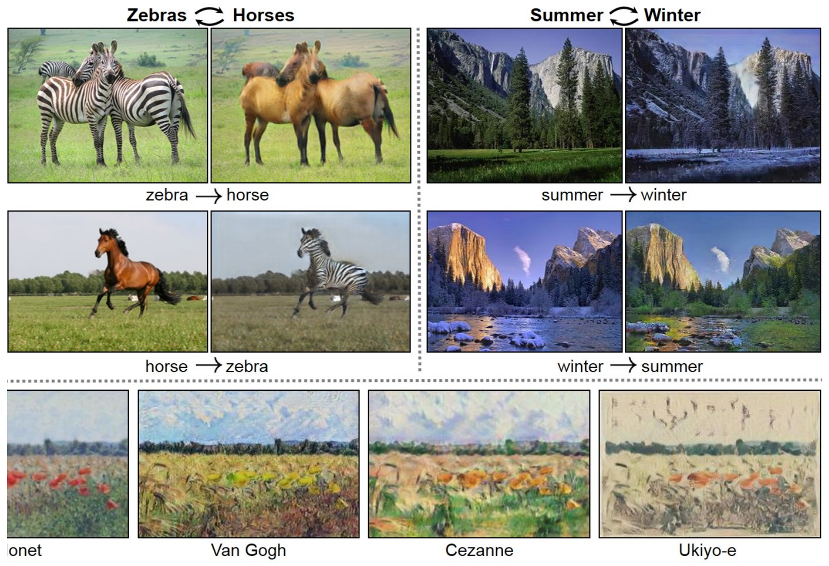
Synthetic Data Generation for Fraud Detection using GANs

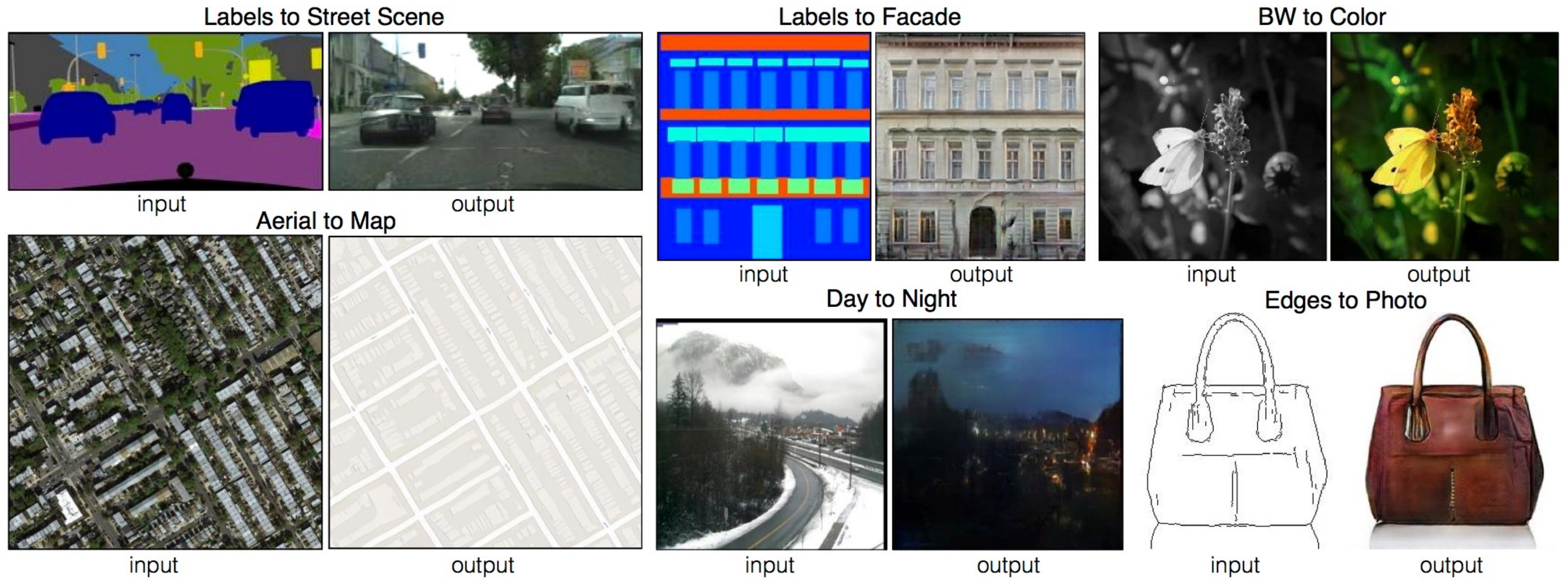
Charitos Charitou
Department of Computer Scie
City, University of Londo
London, UK
charitos.charitou@city.ac.u

Generative Adversarial Networks recover features in astrophysical images of galaxies beyond the deconvolution limit

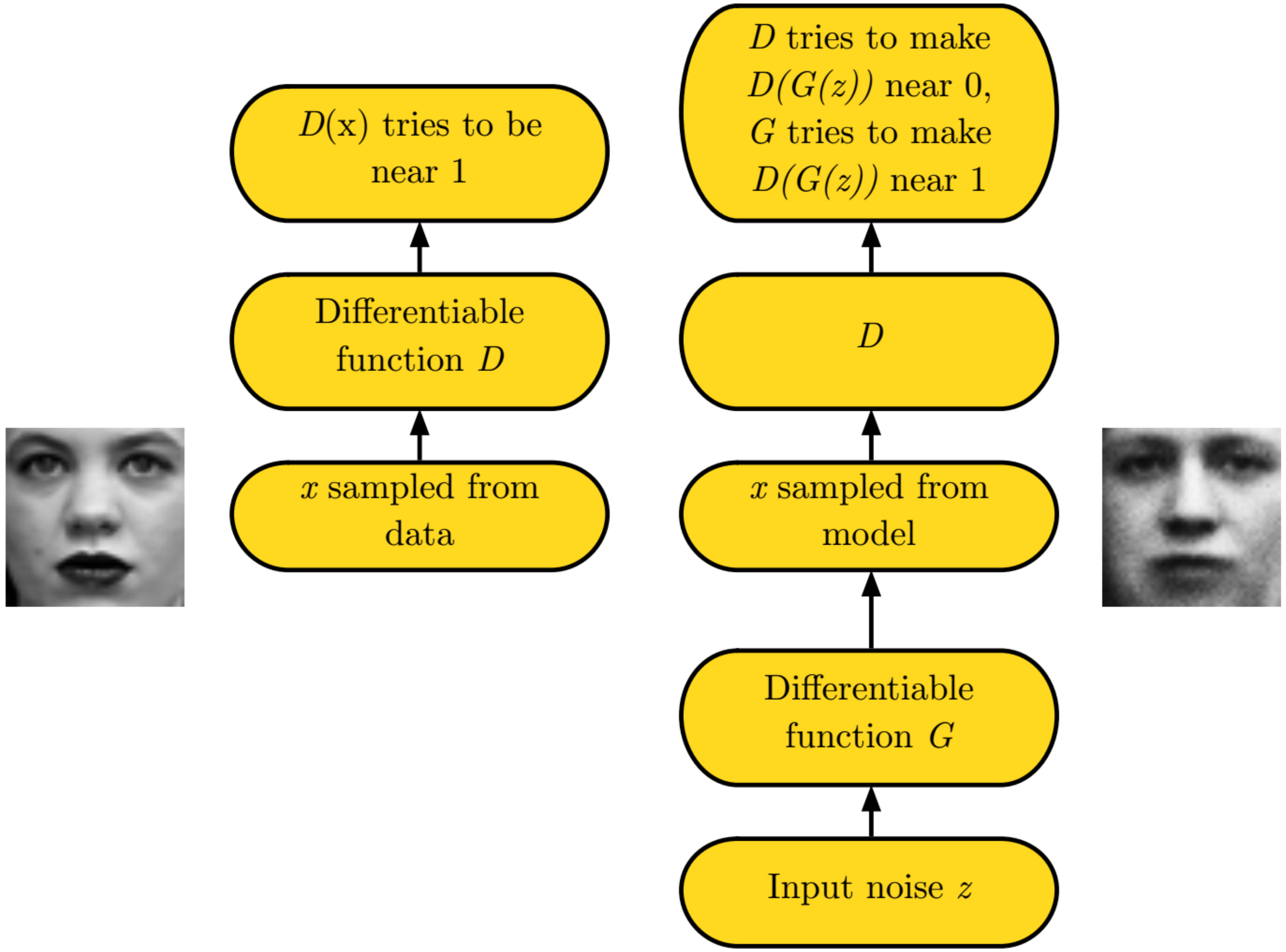
Kevin Schawinski,[1*] Ce Zhang,[2†] Hantian Zhang,[2] Lucas Fowler,[1] and Gokula Krishnan Santhanam[2]

[1]*Institute for Astronomy, Department of Physics, ETH Zurich, Wolfgang-Pauli-Strasse 27, CH-8093, Zürich, Switzerland*
[2]*Systems Group, Department of Computer Science, ETH Zurich, Universitätstrasse 6, CH-8006, Zürich, Switzerland*



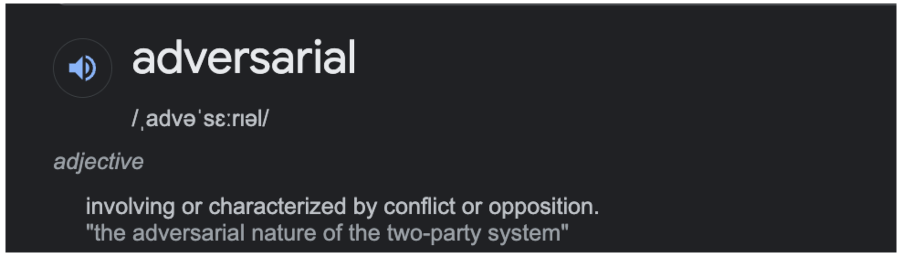


Generative Adversarial Networks


$D(\mathrm{x})$ tries to be near 1
Differentiable function D
x sampled from data
D tries to make $D(G(z))$ near 0, G tries to make $D(G(z))$ near 1
D
x sampled from model
Differentiable function G
Input noise z

NeurIPS 2016 Tutorial: Generative Adversarial Networks

What is a GAN?



adversarial

/ˌadvəˈsɛːrɪəl/

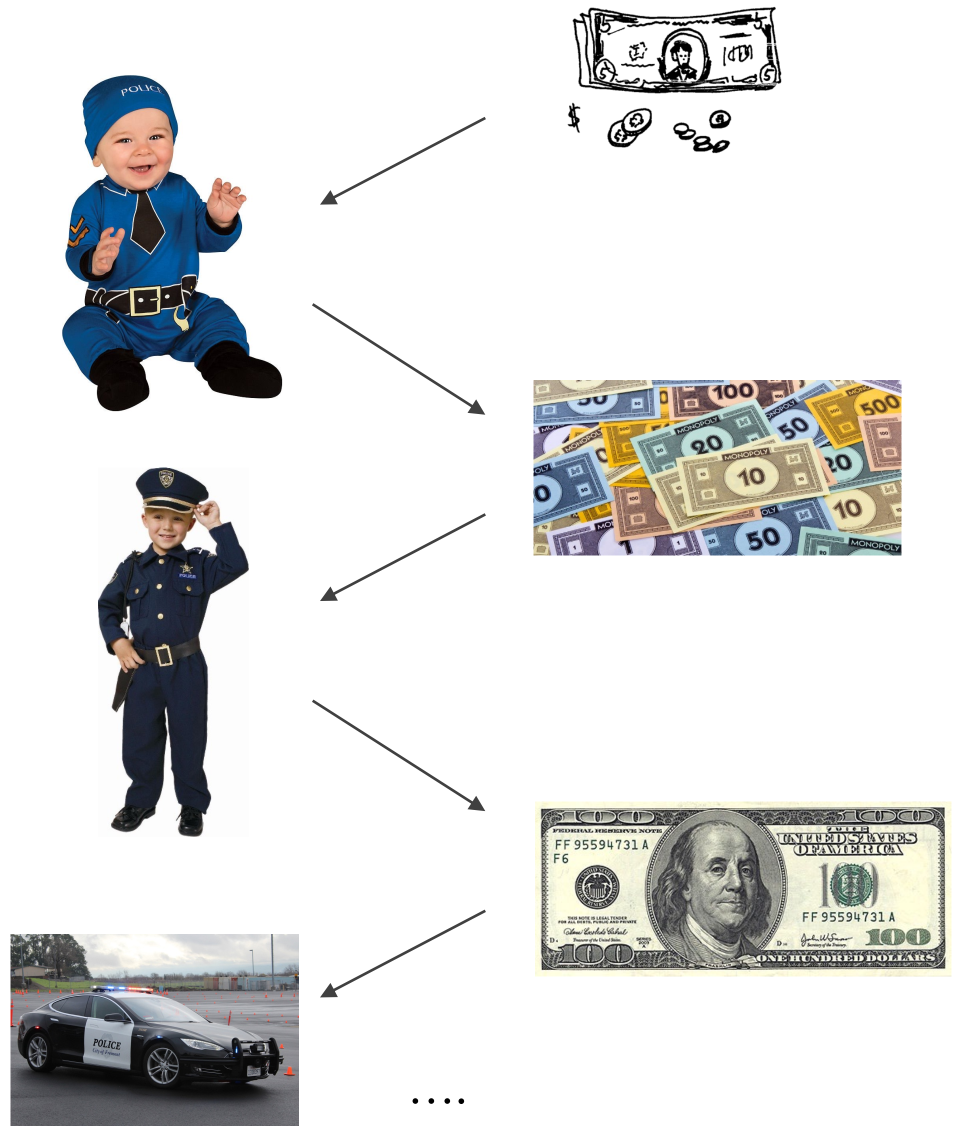
adjective

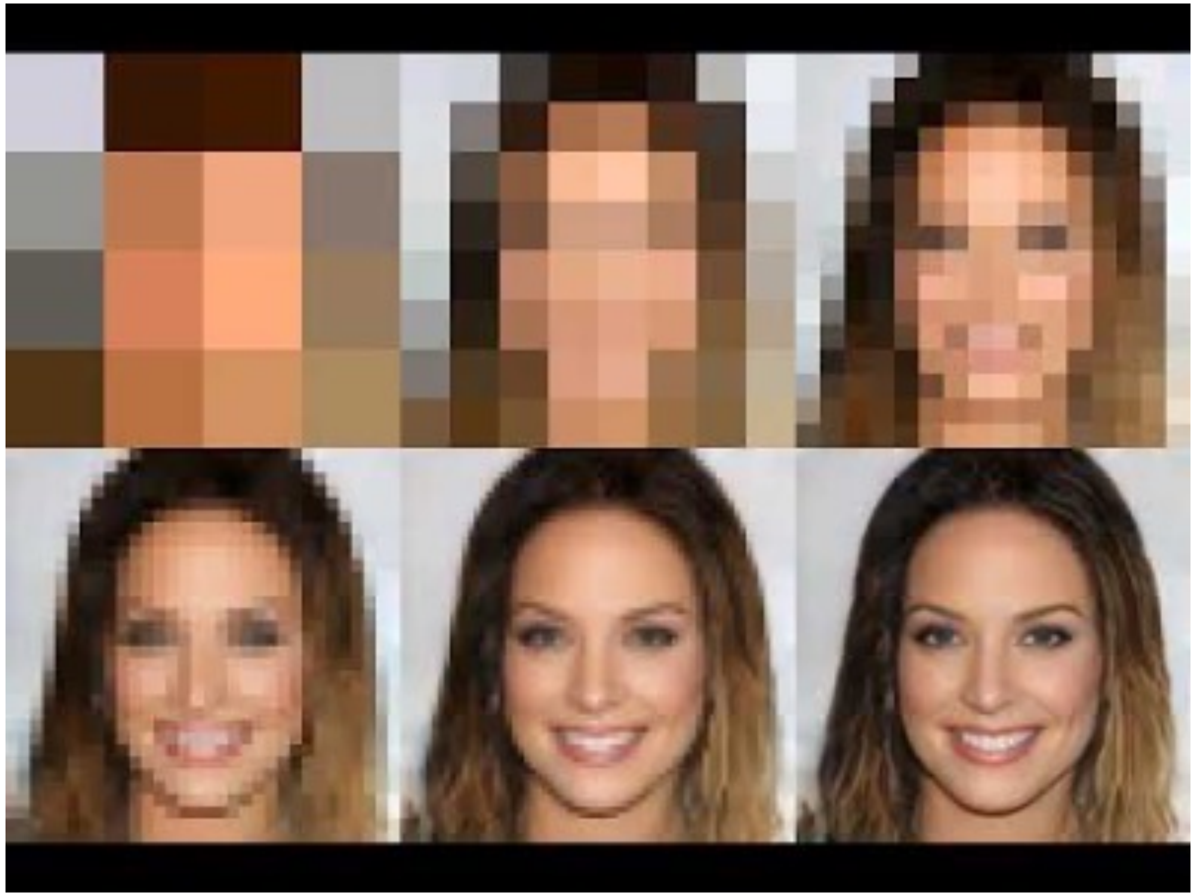
involving or characterized by conflict or opposition.
"the adversarial nature of the two-party system"

- **G**enerative
  - You can sample novel input samples
  - "create" images that never existed
- **A**dversarial
  - Our generative model G learns adversarially
  - by trying to fool an discriminative model D
- **N**etwork
  - Implemented typically as deep neural networks
  - Easy to incorporate new modules
  - Easy to learn via backpropagation

GAN: Intuition: arms race

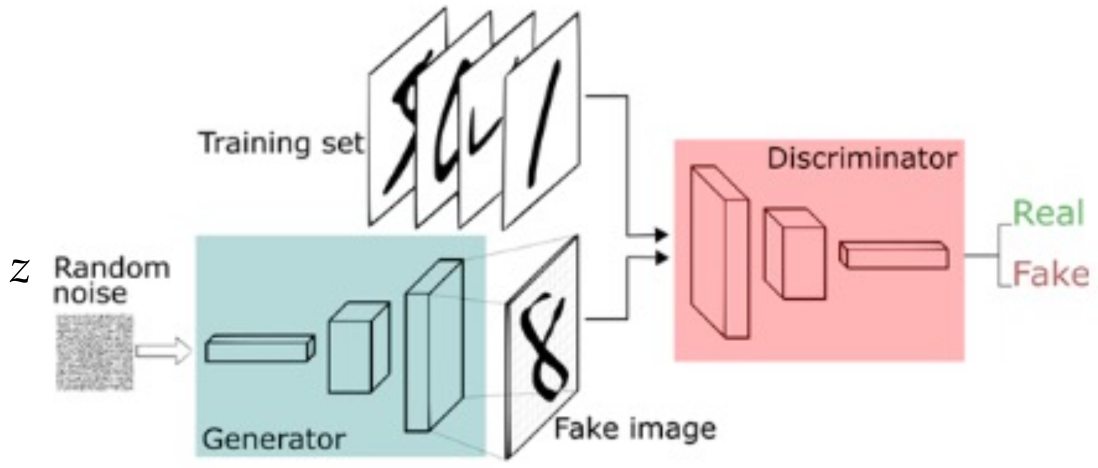
- Police: wants to detect fake money as reliably as possible
- Counterfeiter: wants to make as realistic fake money as possible
- At beginning: both have no clue
- The police forces the counterfeiter to get better as it compares it to real money
  - and vice versa
- Convergent solution ~ Nash equilibrium (game theory)





GAN architecture

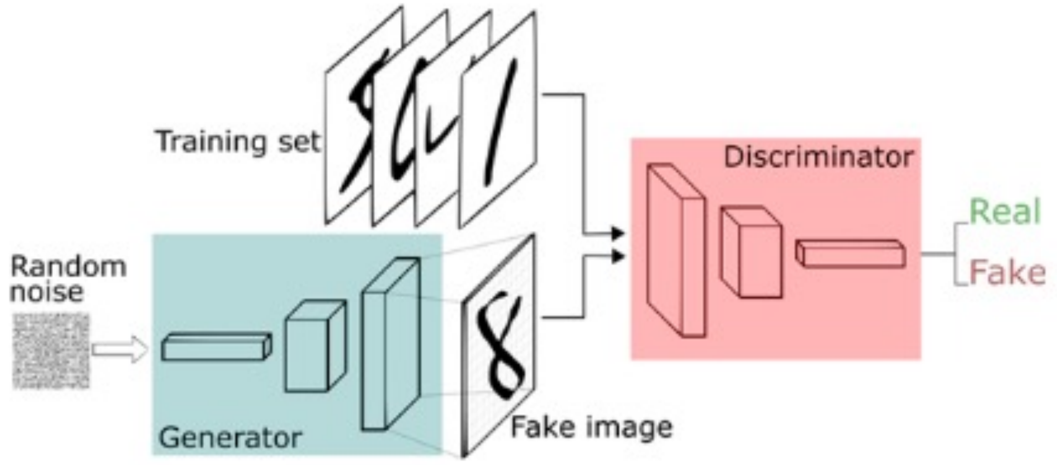
- The GAN comprises two neural networks
  - Generator network $\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta}_{\mathrm{G}})$
  - Discriminator network $\mathrm{y} = D(\boldsymbol{x}; \boldsymbol{\theta}_{\mathrm{D}}) = \begin{cases} +1, \text{if } \mathbf{x} \text{ is predicted 'real'} \\ 0, \text{if } \boldsymbol{x} \text{ is predicted 'fake'} \end{cases}$



GAN has no "encoder" – it's a discriminator

- The "encoder" simply learns features for discriminating between real/fake
  - It's still an "image-in, features/predictions-out" network
- We cannot compute a likelihood of a specific data point (different to VAEs)
- At test time we can only generate new data points

$\boldsymbol{z} \sim \mathcal{N}(0,1)$ or
$\boldsymbol{z} \sim \text{Uniform}(0,1)$
…

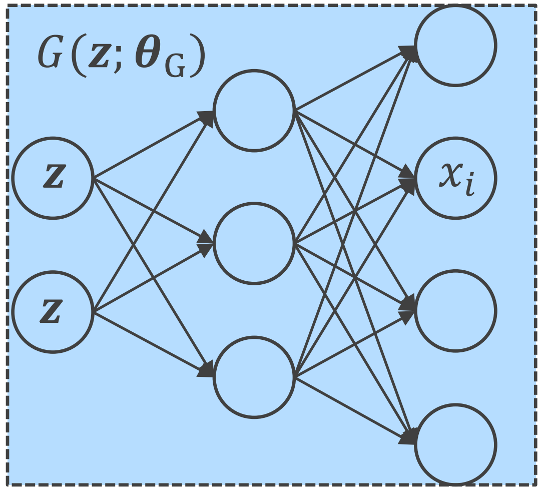


1) Generator network $\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta}_G)$

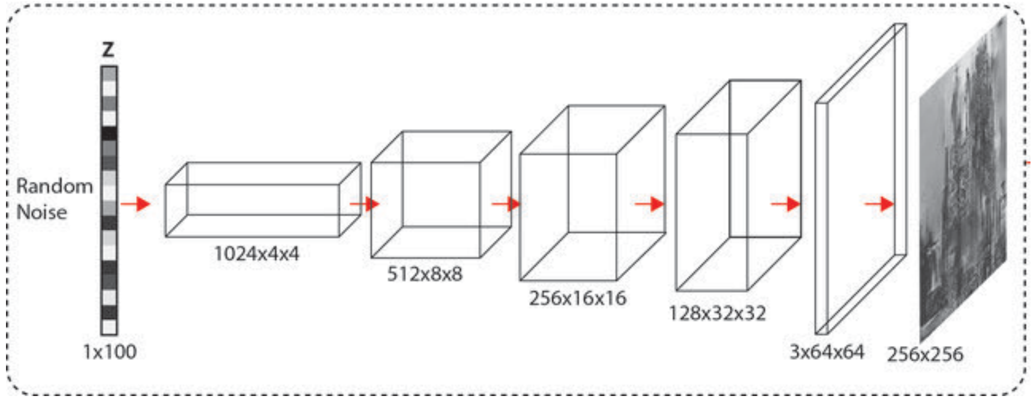
- Any differentiable neural network
- No invertibility requirement → More flexible modelling
- Starts with some random, typically lower dimensional input z
- Various density functions for the noise variable $\boldsymbol{z}$

$$\begin{tikzcd} z \arrow[r] & x \end{tikzcd}$$

$\boldsymbol{z} \sim \mathcal{N}(0,1)$ or
$\boldsymbol{z} \sim \text{Uniform}(0,1)$
$\dots$

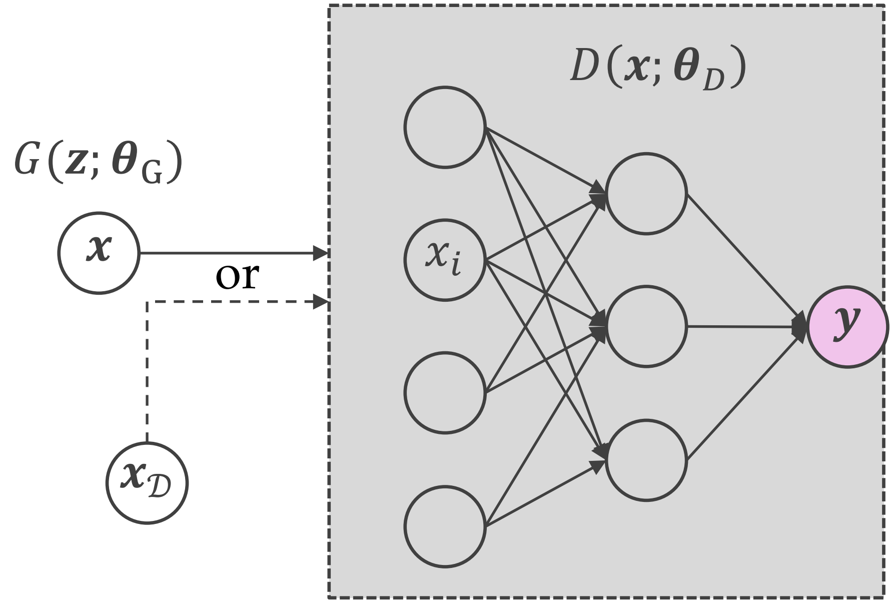


Example architecture



2) Discriminator network $\boldsymbol{y} = D(\boldsymbol{x}; \boldsymbol{\theta}_{\mathrm{D}})$

- Any differentiable neural network
- Receives as inputs
  - *either* real images from the training set
  - or generated images from the generator
  - usually a mix of both in mini-batches
- D must recognize the real from the fake inputs
- The discriminator loss



$D(\boldsymbol{x}; \boldsymbol{\theta}_D)$
$G(\boldsymbol{z}; \boldsymbol{\theta}_{\mathrm{G}})$
$\boldsymbol{x}$
or
x_i
$\boldsymbol{y}$
$\boldsymbol{x}_{\mathcal{D}}$

Binary Cross Entropy loss:

$$l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right],$$

$$J_D(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G) = \frac{1}{2}\mathrm{BCE}(Data, 1) + \frac{1}{2}\mathrm{BCE}(fake, 0)$$

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

$$= -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(x)] - \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{generator}}}[\log(1 - D(\boldsymbol{x}))]$$

Generator & Discriminator: Implementation

- The discriminator is just a standard neural network
- The generator looks like an inverse discriminator (or like a decoder)

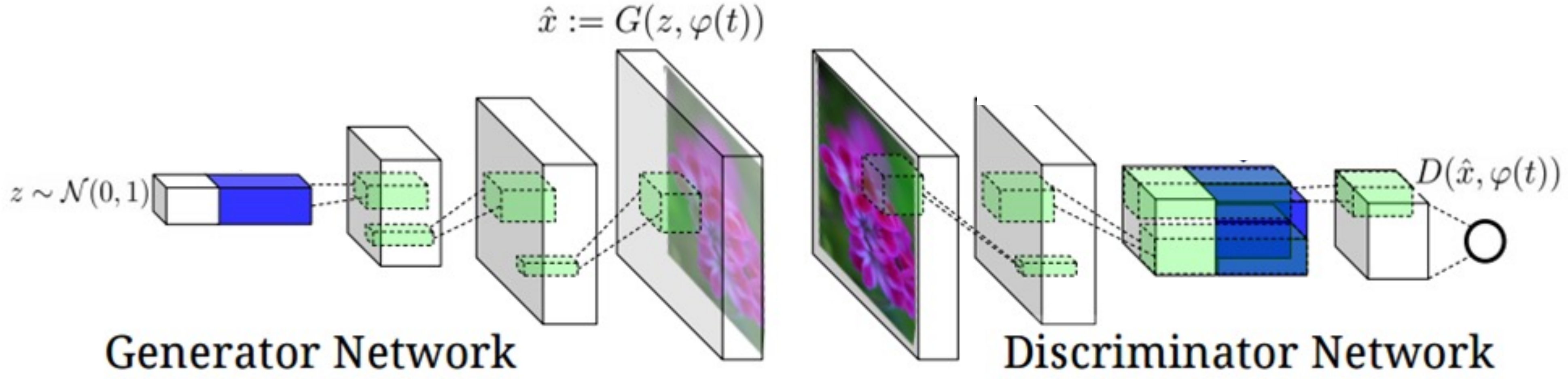


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

Network Architecture

Quiz: The starting point of the GAN is the random noise z. What is true?

1) The noise acts as a regulariser for the Generator
2) Noise models the random variations due to augmentations
3) Even if trained perfectly, not every z that gets samples will produce realistic images
4) A fixed grid of (say) one million points would also work

How do we train the generator?

- Given some generated image, it's not like we have "the equivalent" image in our batch.
- Generator generates some random images independent of comparison batch
- How can we get meaningful gradients?

1) Minimax Loss

- Simplest case: Generator is negative discriminator loss ("zero-sum game")

$$J_G = -J_D$$

  - The lower the generator loss, the higher the discriminator loss
  - Symmetric definitions

- Our learning objective then becomes

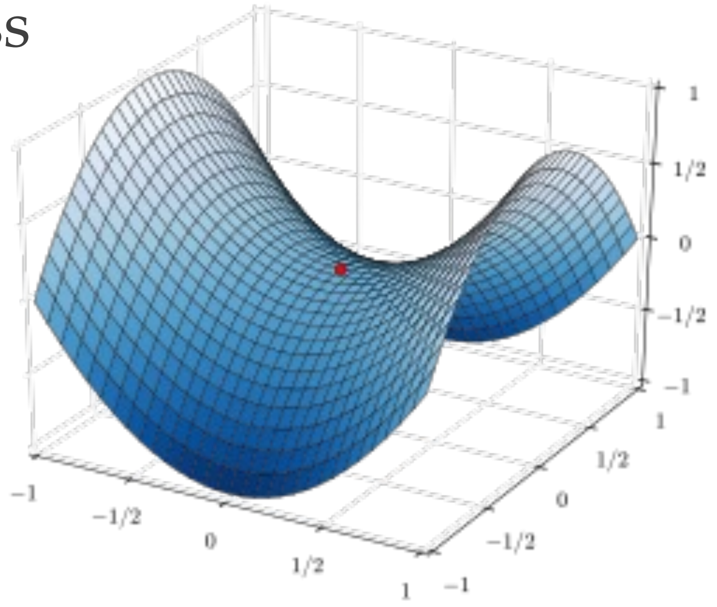
$$V = -J_D(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\theta}_{\mathrm{G}})$$

- $D(\boldsymbol{x}) = 1 \rightarrow$ The discriminator believes that x is a true image
- $D(G(\boldsymbol{z})) = 1 \rightarrow$ The discriminator believes that $G(z)$ is a true image
- So overall loss:

$$\text{Minimize}_G \, \text{Maximize}_D \, J_D$$

1) Minimax Loss

- Learning stops after a while
  - As training iterations increase the discriminator improves: $\frac{dJ_D}{d\boldsymbol{\theta}_\mathrm{D}} \to 0$
  - Then, the generator, preceding the discriminator, vanish
- Equilibrium is a saddle point of the discriminator loss
- This allows for easier theoretical analysis



NeurIPS 2016 Tutorial: Generative Adversarial Networks

2) Heuristic non-saturating loss

- Discriminator loss

$$J_D = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log D(\boldsymbol{x}) \; -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} \log(1 - D(G(\boldsymbol{z})))$$

- Generator loss

$$J_G = -\frac{1}{2}\mathbb{E}_{z\sim p_z} \log(D(G(\boldsymbol{z}))$$

- Equilibrium not any more describable by single loss
  - Discriminator maximizes the log-likelihood of correctly discovering real $\log D(\boldsymbol{x})$ and fake $\log(1 - D(G(\boldsymbol{z})))$ samples
  - The generator $G(\boldsymbol{z})$ maximizes the log-likelihood of the discriminator $\log(D(G(\boldsymbol{z}))$ being wrong.
- Heuristically motivated; generator learns even when discriminator is too good on real images

3) Modifying GANs for max-likelihood

- The discriminator remains the same
- Generator loss

$$J_G = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log(\sigma^{-1}(D\big(G(\boldsymbol{z})\big))$$

- The generator is activated by an inverse sigmoid
  - When discriminator is optimal $\frac{dJ_D}{d\boldsymbol{\theta}_D} \rightarrow 0$
  - the generator gradient matches that of maximum likelihood

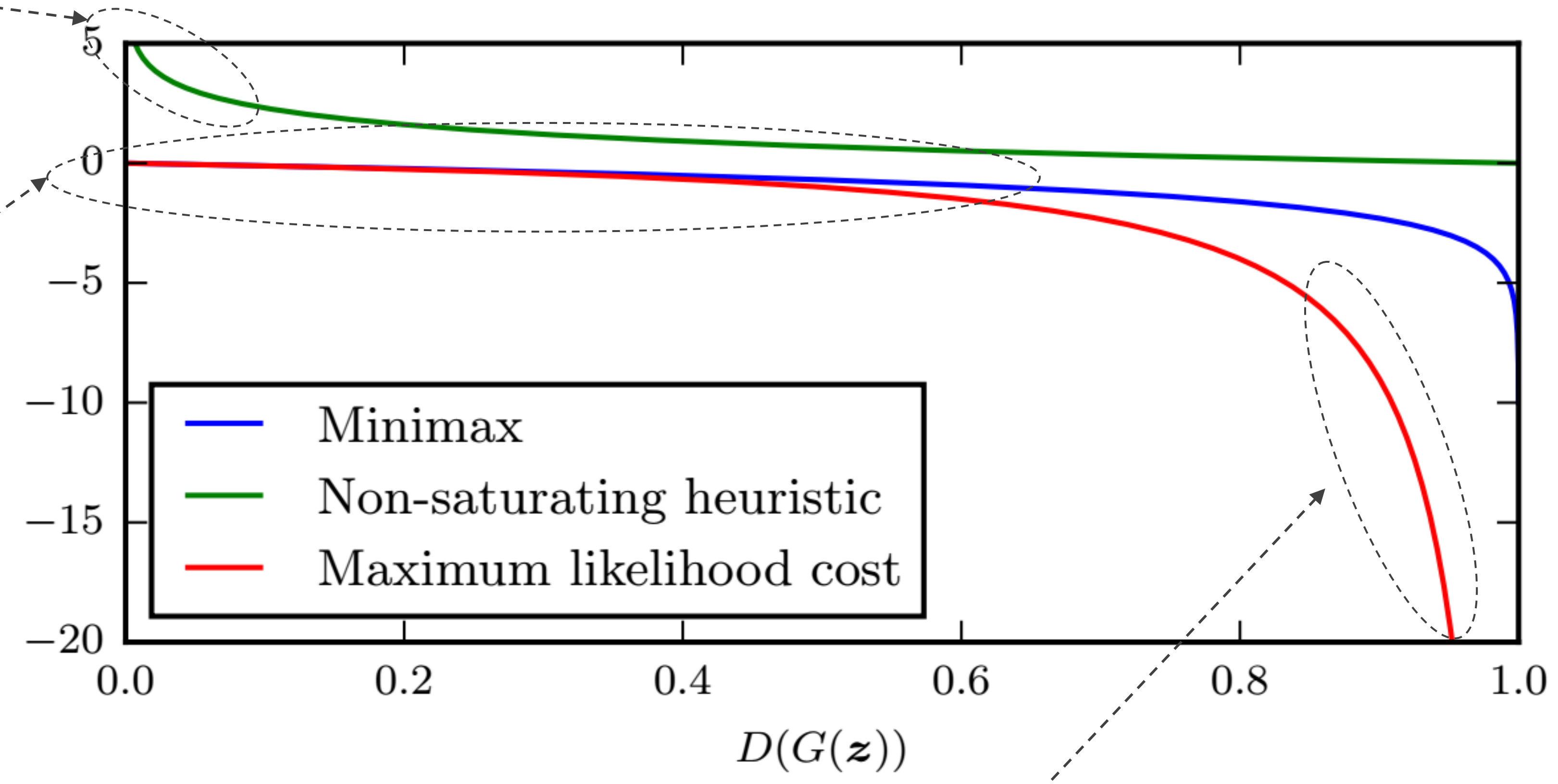
On distinguishability criteria for estimating generative models

Comparison of Generator Losses

$$J_G = -\frac{1}{2}\mathbb{E}_{z\sim p_z}\log(D(G(\boldsymbol{z})))$$

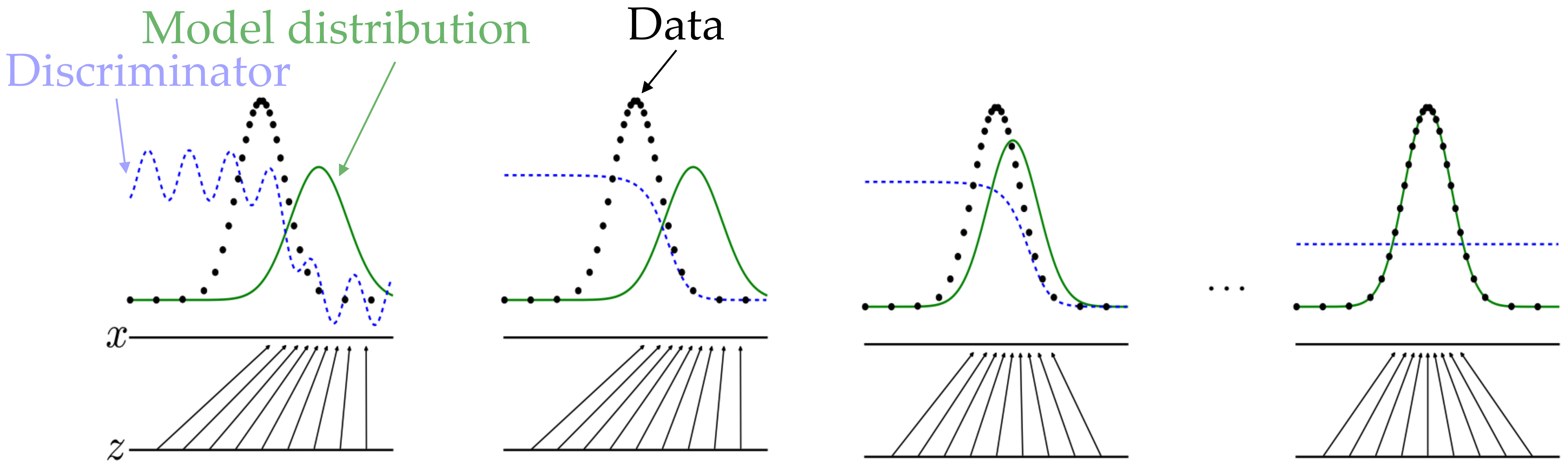
The heuristic loss yields good generator gradients when the discriminator is too good. And smaller gradients as the discriminator gets more confused.

When the discriminator detects fake samples accurately (low $D(G(\boldsymbol{z}))$) the generator has a flat loss curve with both the minimax and the ML losses
→ no gradients in early steps



The ML cost variant generates gradients mostly from the "good generations"
→ all gradients from few samples
→ high variance gradients

Optimal discriminator



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

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

Why is this the optimal discriminator?

- $$L(D,g) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$
  $$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx$$
  - Minimize $\mathcal{L}(D,G)$ w.r.t. $D \rightarrow \frac{d\mathcal{L}}{dD} = 0$ and ignore the integral
  - The function $x \rightarrow a \log x + b \log(1-x)$ attains max in $[0,1]$ at $\frac{a}{a+b}$
- The optimal discriminator
  $$D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$
  - And at **optimality** $p_g(\boldsymbol{x}) \rightarrow p_{\text{data}}(\boldsymbol{x})$, thus
  $$D^*(\boldsymbol{x}) = \frac{1}{2}$$
  $$L(G^*, D^*) = -2\log 2$$

Great blog: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

GANs and Jensen-Shannon divergence

- Expanding the JS divergence for the optimal discriminator $D^*(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x})+p_g(\boldsymbol{x})}$

$$D_{JS}(p_r||p_g) := \frac{1}{2} D_{KL}(p_r||\frac{p_r+p_g}{2}) + \frac{1}{2} D_{KL}(p_g||\frac{p_r+p_g}{2})$$

$$= \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x)+p_g(x)}\, dx + \log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r(x)+p_g(x)}\, dx\right)$$

$$= \frac{1}{2}(\log 4 + L(G, D^*))$$

- So, $L(G, D^*) = 2D_{JS}(p_r \parallel p_g) - 2\log 2$
  - And we just found out that $L(G^*, D^*) \geq -2\log 2$
  - $\rightarrow$ for $L(G^*, D^*) = -2\log 2 \Rightarrow D_{JS}(p_r||p_g) = 0$

https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Is the divergence important?

- Does the divergence make a difference?
- Is there a difference between KL-divergence, Jensen-Shannon divergence, …

$$D_{KL}(p_r||p_g) = \int_x p_r \log \frac{p_r}{p_g} dx$$

$$D_{JS}(p_r||p_g) = \frac{1}{2} D_{KL}(p_r|| \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g|| \frac{p_r + p_g}{2})$$

KL vs JS

- JS is symmetric
- KL is not

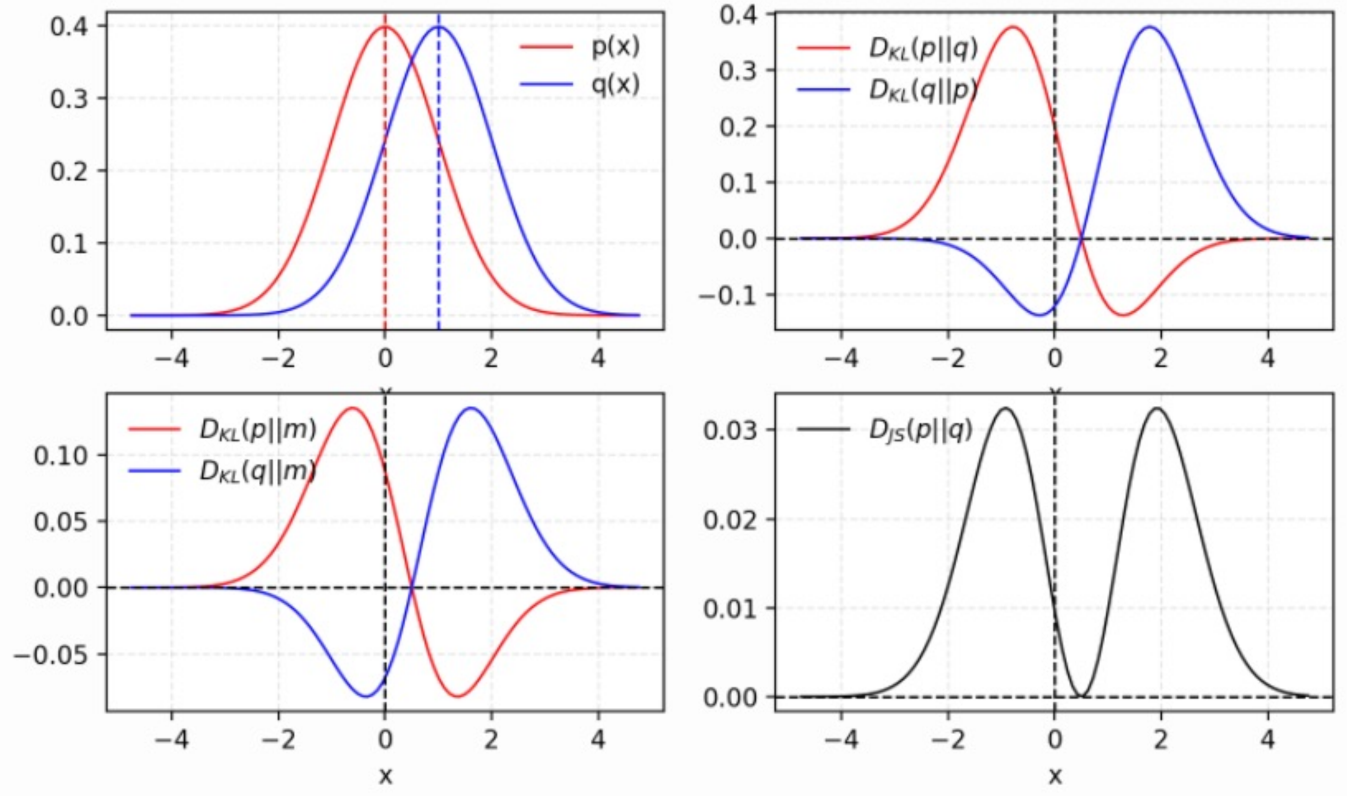
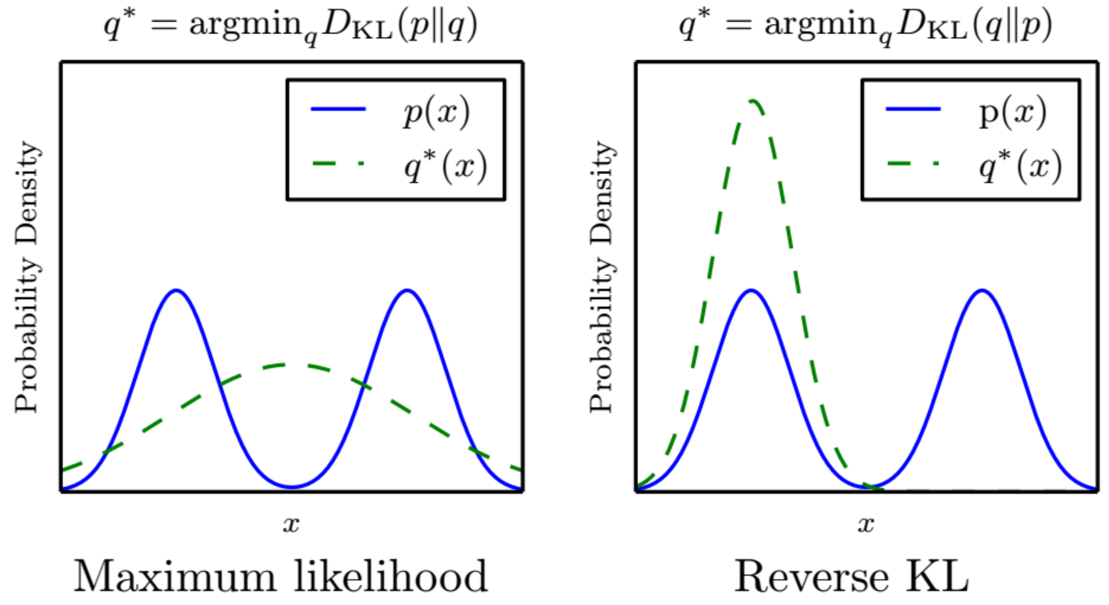


Fig. 1. Given two Gaussian distribution, p with mean=0 and std=1 and q with mean=1 and std=1. The average of two distributions is labelled as $m = (p + q)/2$. KL divergence D_{KL} is asymmetric but JS divergence D_{JS} is symmetric.

https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Is the divergence important?

- $D_{KL}(p(x)||q^*(x)) \rightarrow$ high probability everywhere that the data occurs
- $D_{KL}(q^*(x)||p(x)) \rightarrow$ low probability wherever the data does not occur
- Backward KL is 'zero forcing' the learned model → makes model "conservative"
  - $D_{KL}(q^*(x)||p(x)) = \int q^*(x) \log \frac{q^*(x)}{p(x)}$
  - $q^*(x) \rightarrow 0$ where $\frac{q^*(x)}{p(x)}$ cannot be good
  - Avoid areas where $p(x) \rightarrow 0$
  - "mode seeking"



p_r is what we get from our data and cannot change
p_g is what we learn with our model

General observations

- GANs are designed to deliver good generations
- With GANs we do not write down the densities explicitly
  - We can do generations
  - But cannot easily do inferences, compute conditionals or marginals

Training procedure

- Use SGD-like algorithm of choice
  - Adam optimizer is a good choice
- Use two mini-batches simultaneously
  - The first mini-batch contains real examples from the training set
  - The second mini-batch contains fake generated examples from the generator
- Optional: run k-steps of one network (e.g. discriminator) for every step of the other one (e.g. generator), many heuristics exist here.

How research gets done part 8

Katalin Karikó

Previous parts:
[fundamental understanding/read papers, how-to-read-papers, implement & tinker with code, realise and seek *funny* moments, MVP/principles/benchmarks/baselines, when to (not) give up/impact-vs-work]

Today:

- Ok, so you didn't give up and you're on to something non-trivial.
- Next: How do you show/ analyze what's happening or why your method is better?
- Answer: Ablations
  - The key idea is to "only vary one thing at a time"
  - (Same principle behind when designing experiments in the investigation phase!)
  - Never change two things at the same time, you won't know if it was A or B that helped
  - Some examples:
- Show simple, easy to understand cases (sometimes toy examples)
- **One idea per paper!**

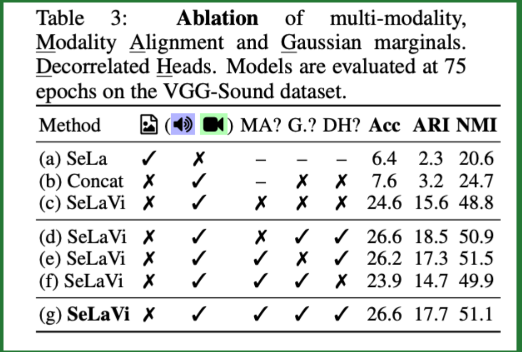
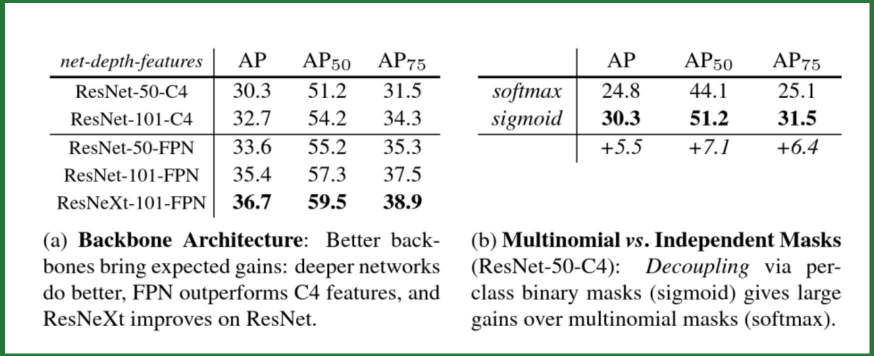


Table 3: **Ablation** of multi-modality, Modality Alignment and Gaussian marginals. Decorrelated Heads. Models are evaluated at 75 epochs on the VGG-Sound dataset.

Method	🖼	(🔊 🎥)	MA?	G.?	DH?	**Acc**	**ARI**	**NMI**
(a) SeLa	✓	✗	–	–	–	6.4	2.3	20.6
(b) Concat	✗	✓	–	✗	✗	7.6	3.2	24.7
(c) SeLaVi	✗	✓	✗	✗	✗	24.6	15.6	48.8
(d) SeLaVi	✗	✓	✗	✓	✓	26.6	18.5	50.9
(e) SeLaVi	✗	✓	✓	✗	✓	26.2	17.3	51.5
(f) SeLaVi	✗	✓	✓	✓	✗	23.9	14.7	49.9
(g) **SeLaVi**	✗	✓	✓	✓	✓	26.6	17.7	51.1



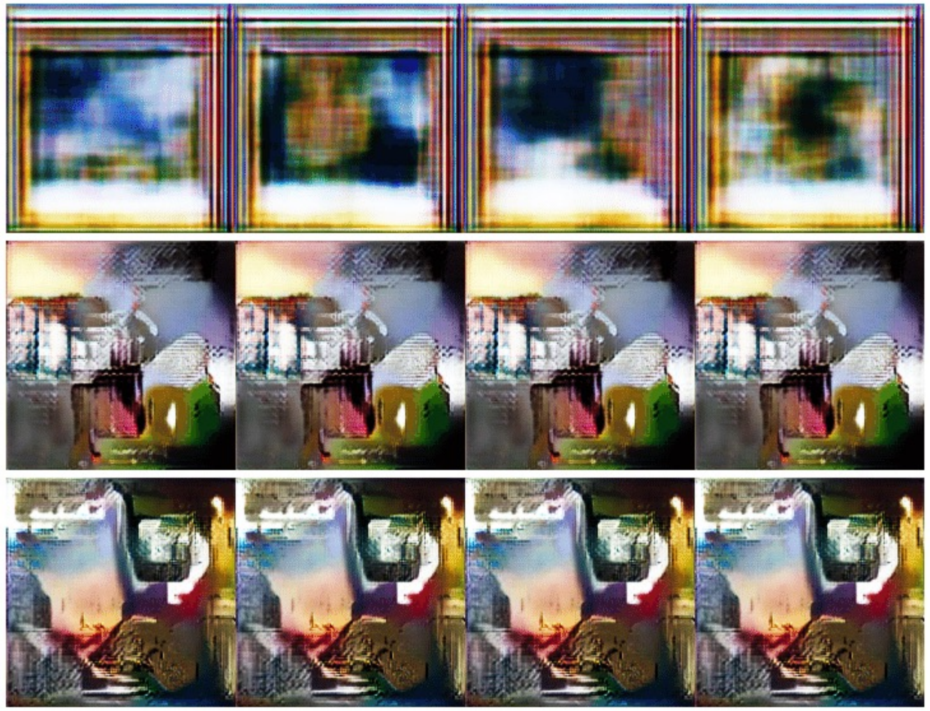
net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

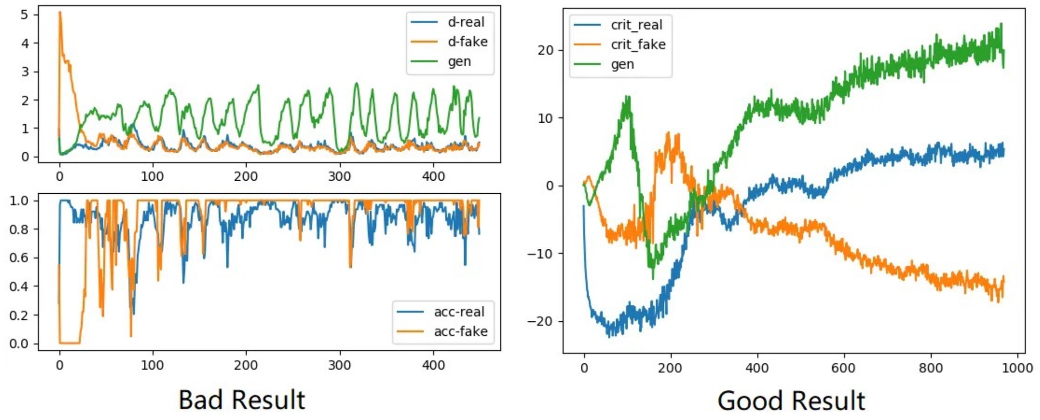
(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

Challenges




d-real
d-fake
gen
acc-real
acc-fake
Bad Result
crit_real
crit_fake
gen
Good Result

Challenge 1: Vanishing Gradients

- If the discriminator is quite bad
  → the generator gets confused
  → no reasonable generator gradients
- If the discriminator is near perfect
  → gradients go to 0, no learning anymore
- Bad if early in the training
  - Easier to train the discriminator than generator

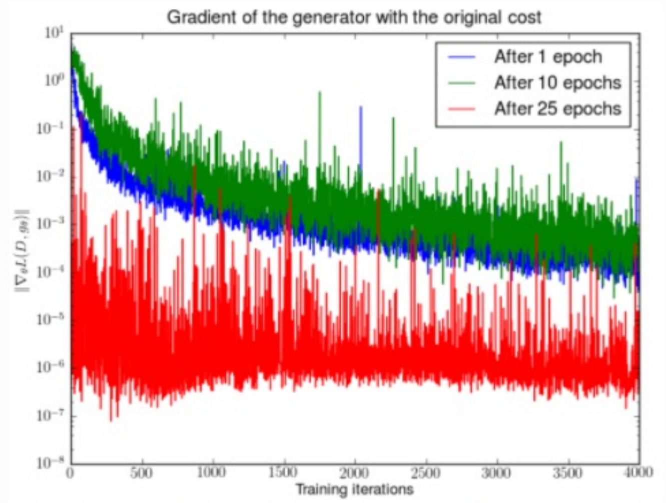
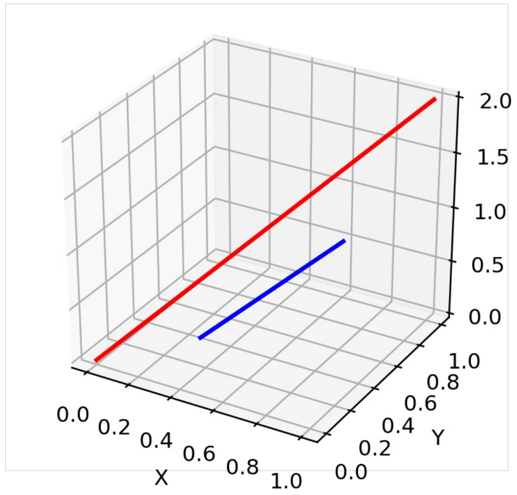
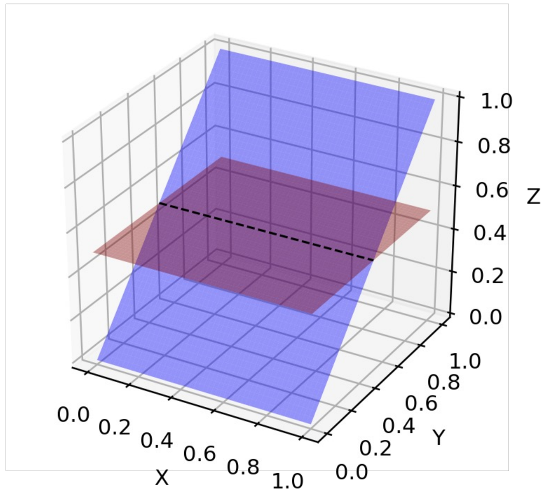


Fig. 5. First, a DCGAN is trained for 1, 10 and 25 epochs. Then, with the ***generator fixed****, a discriminator is trained from scratch and measure the gradients with the original cost function. We see the gradient norms* ***decay quickly*** *(in log scale), in the best case 5 orders of magnitude after 4000 discriminator iterations. (Image source: Arjovsky and Bottou, 2017)*

Challenge 2: Low dimensional supports

- Data lie in low-dim manifolds
- However, the manifold is not known
- During training p_g is not perfect either, especially at the start
- So, the support of p_r and p_g is non-overlapping and disjoint
  $\rightarrow$ not good for KL/JS divergences
- Easy to find a discriminating line



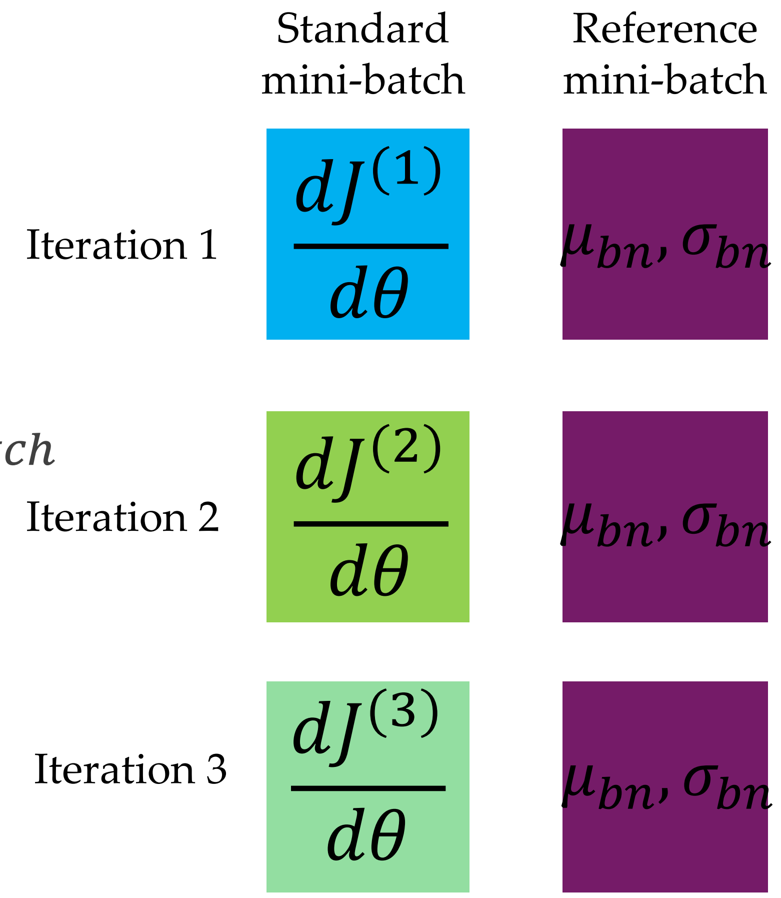


Challenge 3: Batch Normalization

- Batch-normalization causes strong intra-batch correlation
  - Activations depend on other inputs
  - → Generations depend on other inputs
- Generations look smooth but awkward

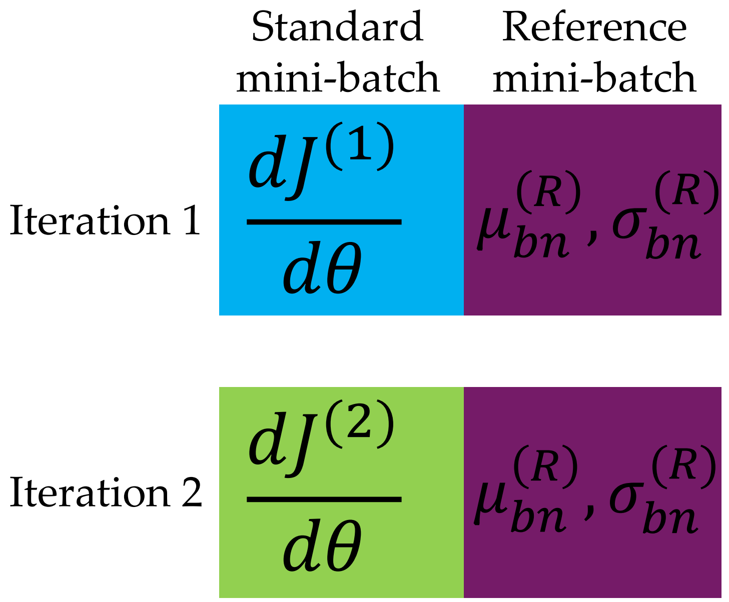
Reference batch normalization

- Training with two mini-batches
- Fixed reference mini-batch to compute $\mu_{bn}^{ref}, \sigma_{bn}^{ref}$
- Second mini-batch x_{batch} for training
- Same training, only use $\mu_{bn}^{ref}, \sigma_{bn}^{ref}$ to normalize x_{batch}
- Problem: Overfitting to the reference mini-batch



Virtual batch normalization

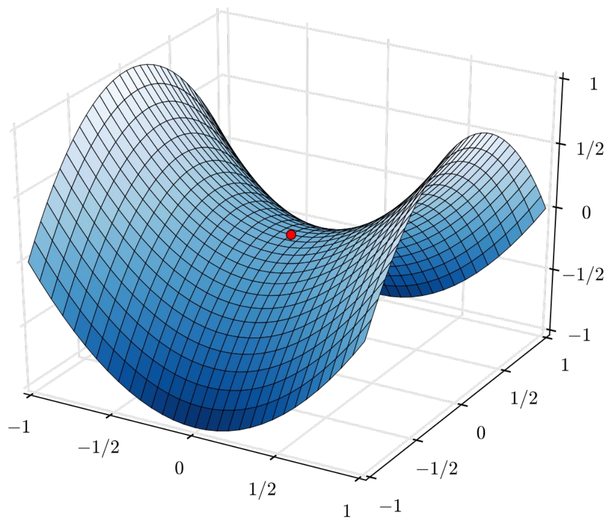
- Append the reference batch to regular mini-batch
- GPU memory is a potential issue


Standard mini-batch
Reference mini-batch
Iteration 1
$\frac{dJ^{(1)}}{d\theta}$
$\mu_{bn}^{(R)}, \sigma_{bn}^{(R)}$
Iteration 2
$\frac{dJ^{(2)}}{d\theta}$
$\mu_{bn}^{(R)}, \sigma_{bn}^{(R)}$

- OR: simply don't use BatchNorm! $\rightarrow$e.g. StyleGAN uses InstanceNorm
- Adaptive InstanceNorm $AdaIN(x,y) = \sigma(y)(\frac{x-\mu(x)}{\sigma(x)}) + \mu(y)$

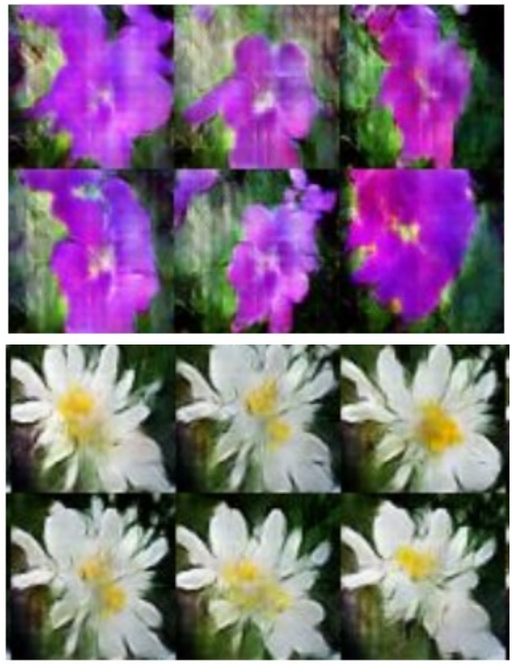
Challenge 4: Convergence

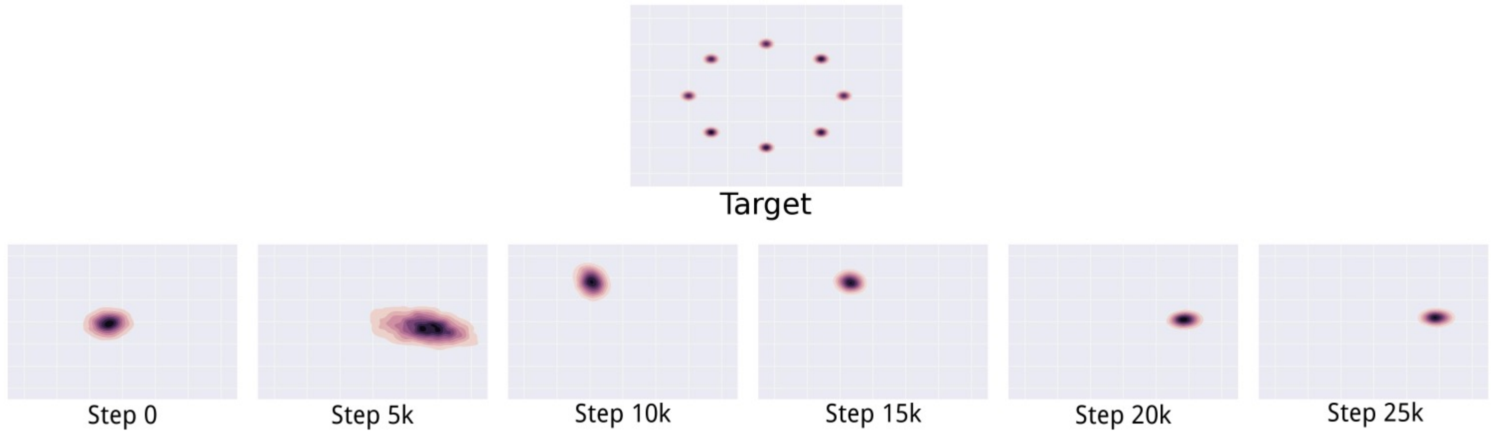
- Optimization is tricky and unstable
  - finding a saddle point does not imply a global minimum
  - A saddle point is also sensitive to disturbances
- An equilibrium might not even be reached (models can train for weeks)
- Mode-collapse is the most severe form of non-convergence

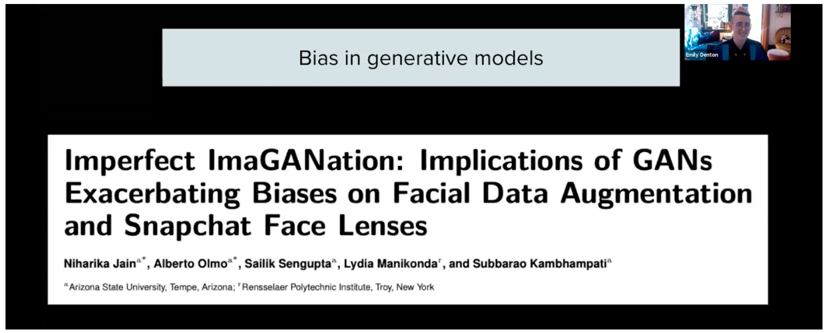


Challenge: mode collapse

- Discriminator converges to the correct distribution
- Generator however places all mass in the most likely point
- All other modes are ignored
  - Underestimating variance
- Low diversity in generating samples






Bias in generative models

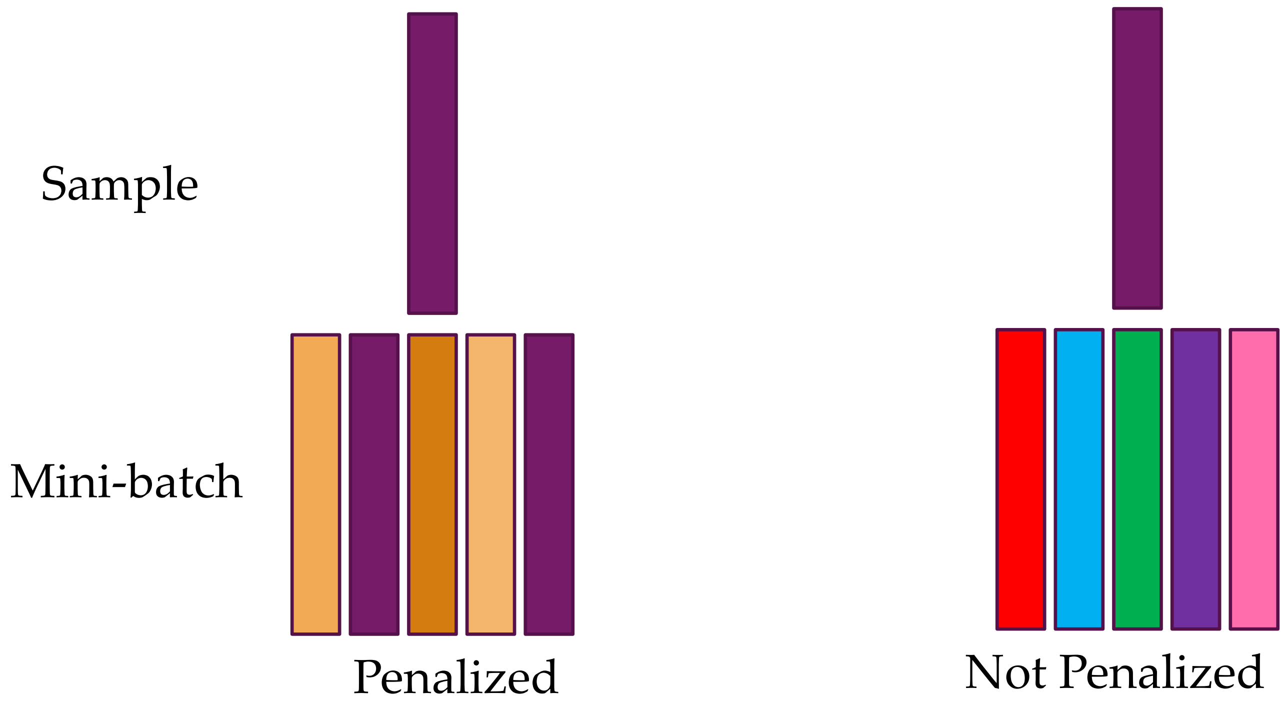
Imperfect ImaGANation: Implications of GANs Exacerbating Biases on Facial Data Augmentation and Snapchat Face Lenses

Niharika Jain[a,*], Alberto Olmo[a,*], Sailik Sengupta[a], Lydia Manikonda[r], and Subbarao Kambhampati[a]

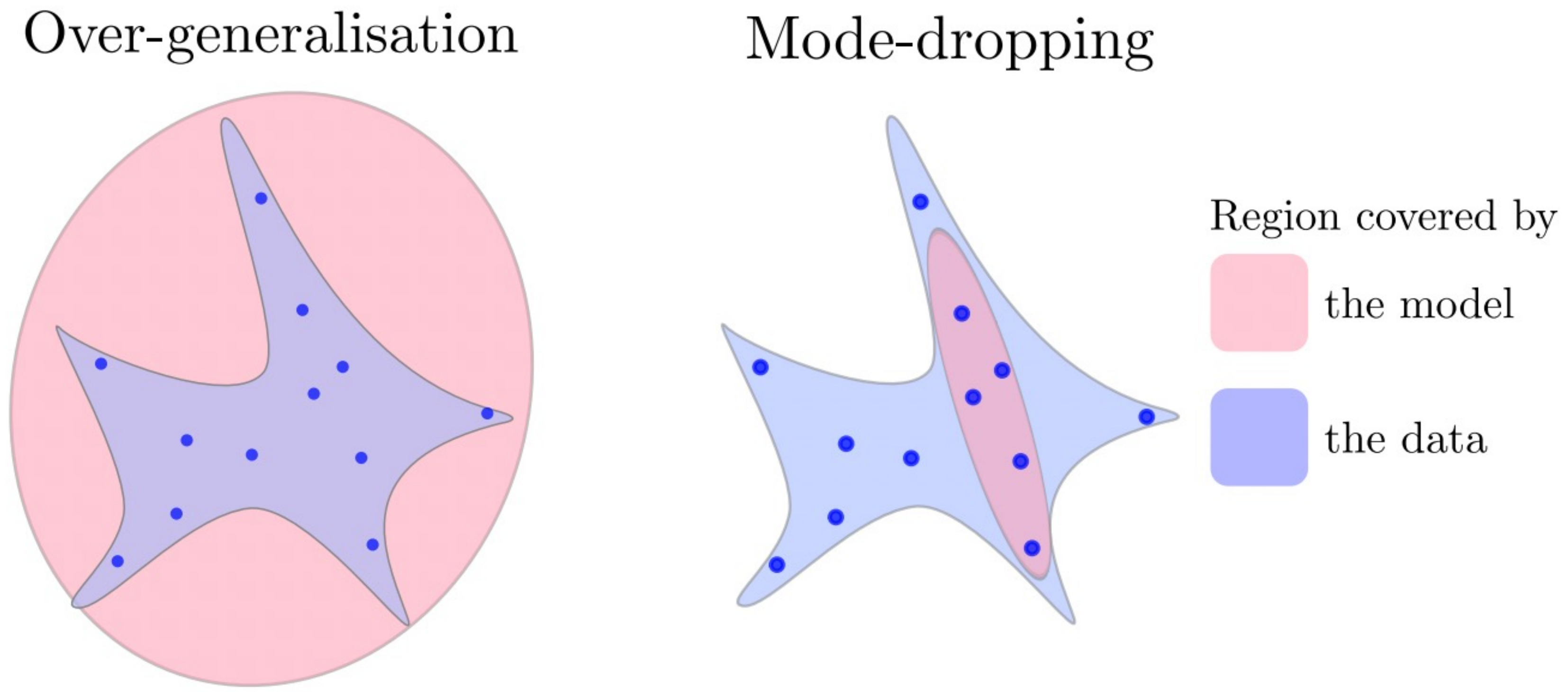
[a]Arizona State University, Tempe, Arizona; [r]Rensselaer Polytechnic Institute, Troy, New York

Potential solution regularize for diversity

- Classify each sample by comparing to other examples in the mini-batch
- If samples are too similar, the model is penalized

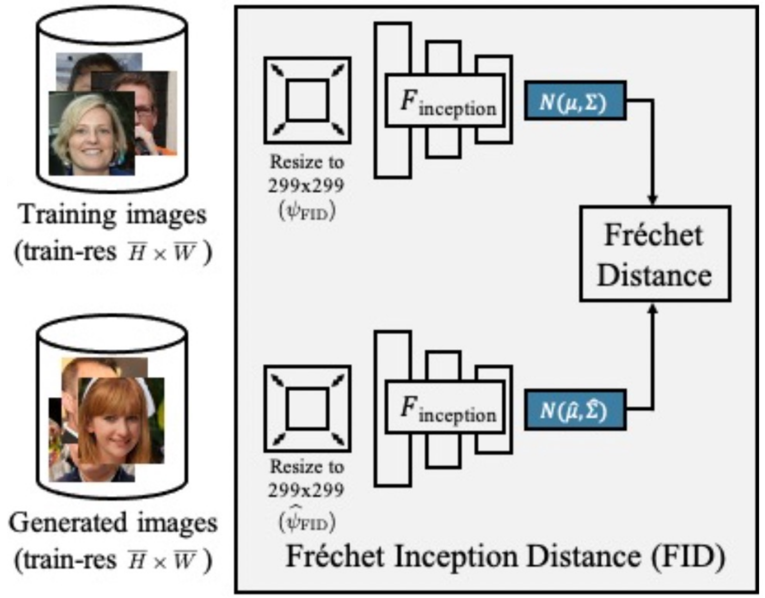


Mode-collapse vs over-generalisation



Challenge: how to evaluate?

FID



Training images
(train-res $\overline{H} \times \overline{W}$)
Generated images
(train-res $\overline{H} \times \overline{W}$)
Resize to 299x299 (ψ_{FID})
Resize to 299x299 ($\widehat{\psi}_{FID}$)
$F_{inception}$
$N(\mu, \Sigma)$
$N(\hat{\mu}, \hat{\Sigma})$
Fréchet Distance
Fréchet Inception Distance (FID)

- It would be nice to quantitatively evaluate the model
- FID, but some problems remain
- For GANs it is hard to estimate the likelihood
- In the absence of a precise evaluation metric, do GANs do truly good generations or generations that appeal/fool to the human eye?
  - Can we trust the generations for critical applications, like medical tasks?
  - *'Are humans a good discriminator for the converged generator?'*

Challenge: beyond images

- The generator must be differentiable
- Tasks with discrete outputs (like text) are ruled out
  - modifications are necessary to flow gradients through discrete variables
- Similarly for other types of structured data like graphs

Some open challenges for GANs

- What are the trade-offs between GANs and other generative models?
  - Speed vs accurate generation etc.
- What sorts of distributions can GANs model?
- What can we say about the global convergence of the training dynamics?
- How should we evaluate GANs and when should we use them?
- GAN scaling: dataset size and model size
- GANs and adversarial examples?

One-sided label smoothing

- Default discriminator cost:

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

- One-sided label smoothing:

```
cross_entropy(0.9, discriminator(data))
+ cross_entropy(0., discriminator(faked))
```

- Do not smooth negative labels:

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

Benefits of label smoothing

- Mitigate overconfidence
- Does not reduce classification accuracy, only confidence
- Specifically for GANs
  - Prevents too high gradients to Generator
  - Prevents extrapolating to encourage extreme samples
- General strategy for improving models (e.g. because there’s noise in annotations and it ensures a more smooth embedding space)

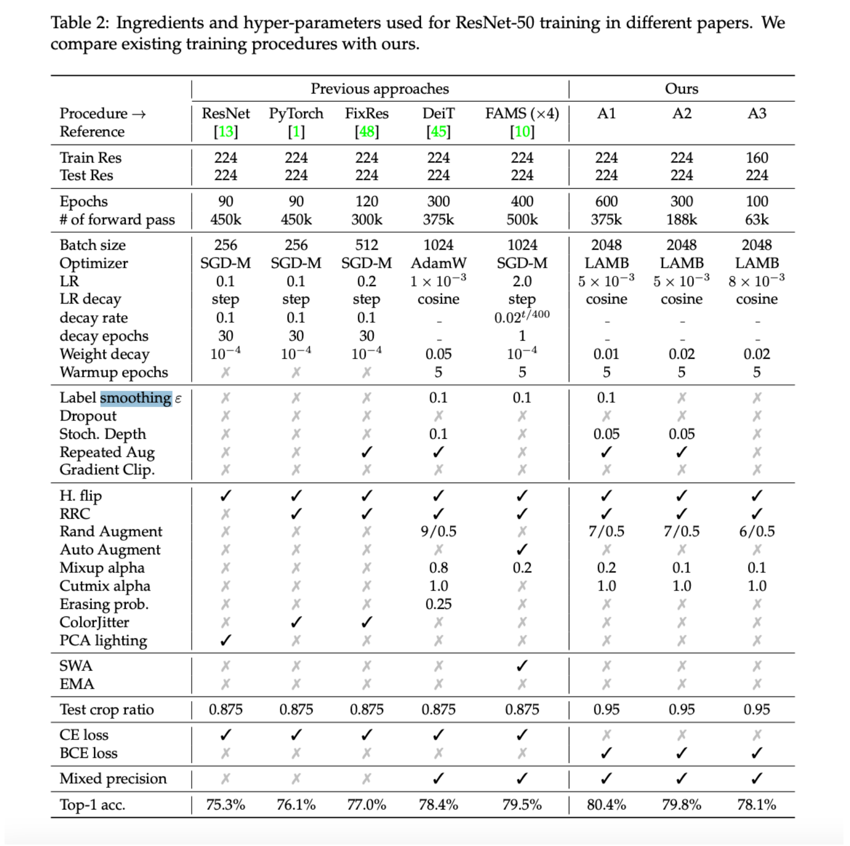
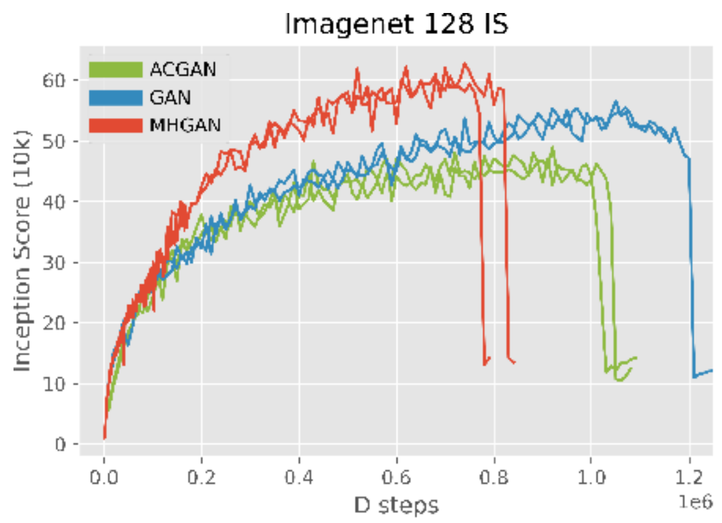


Table 2: Ingredients and hyper-parameters used for ResNet-50 training in different papers. We compare existing training procedures with ours.

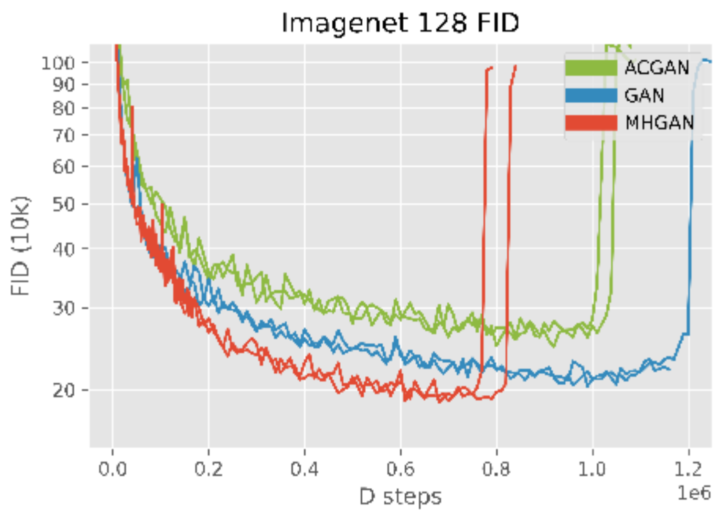
	Previous approaches					Ours		
Procedure → Reference	ResNet [13]	PyTorch [1]	FixRes [48]	DeiT [45]	FAMS ($\times 4$) [10]	A1	A2	A3
Train Res	224	224	224	224	224	224	224	160
Test Res	224	224	224	224	224	224	224	224
Epochs	90	90	120	300	400	600	300	100
# of forward pass	450k	450k	300k	375k	500k	375k	188k	63k
Batch size	256	256	512	1024	1024	2048	2048	2048
Optimizer	SGD-M	SGD-M	SGD-M	AdamW	SGD-M	LAMB	LAMB	LAMB
LR	0.1	0.1	0.2	1×10^{-3}	2.0	5×10^{-3}	5×10^{-3}	8×10^{-3}
LR decay	step	step	step	cosine	step	cosine	cosine	cosine
decay rate	0.1	0.1	0.1	-	$0.02^{t/400}$	-	-	-
decay epochs	30	30	30	-	1	-	-	-
Weight decay	10^{-4}	10^{-4}	10^{-4}	0.05	10^{-4}	0.01	0.02	0.02
Warmup epochs	✗	✗	✗	5	5	5	5	5
Label smoothing ε	✗	✗	✗	0.1	0.1	0.1	✗	✗
Dropout	✗	✗	✗	✗	✗	✗	✗	✗
Stoch. Depth	✗	✗	✗	0.1	✗	0.05	0.05	✗
Repeated Aug	✗	✗	✓	✓	✗	✓	✓	✗
Gradient Clip.	✗	✗	✗	✗	✗	✗	✗	✗
H. flip	✓	✓	✓	✓	✓	✓	✓	✓
RRC	✗	✓	✓	✓	✓	✓	✓	✓
Rand Augment	✗	✗	✗	9/0.5	✗	7/0.5	7/0.5	6/0.5
Auto Augment	✗	✗	✗	✗	✓	✗	✗	✗
Mixup alpha	✗	✗	✗	0.8	0.2	0.2	0.1	0.1
Cutmix alpha	✗	✗	✗	1.0	✗	1.0	1.0	1.0
Erasing prob.	✗	✗	✗	0.25	✗	✗	✗	✗
ColorJitter	✗	✓	✓	✗	✗	✗	✗	✗
PCA lighting	✓	✗	✗	✗	✗	✗	✗	✗
SWA	✗	✗	✗	✗	✓	✗	✗	✗
EMA	✗	✗	✗	✗	✗	✗	✗	✗
Test crop ratio	0.875	0.875	0.875	0.875	0.875	0.95	0.95	0.95
CE loss	✓	✓	✓	✓	✓	✗	✗	✗
BCE loss	✗	✗	✗	✗	✗	✓	✓	✓
Mixed precision	✗	✗	✗	✓	✓	✓	✓	✓
Top-1 acc.	75.3%	76.1%	77.0%	78.4%	79.5%	80.4%	79.8%	78.1%

Generally often used

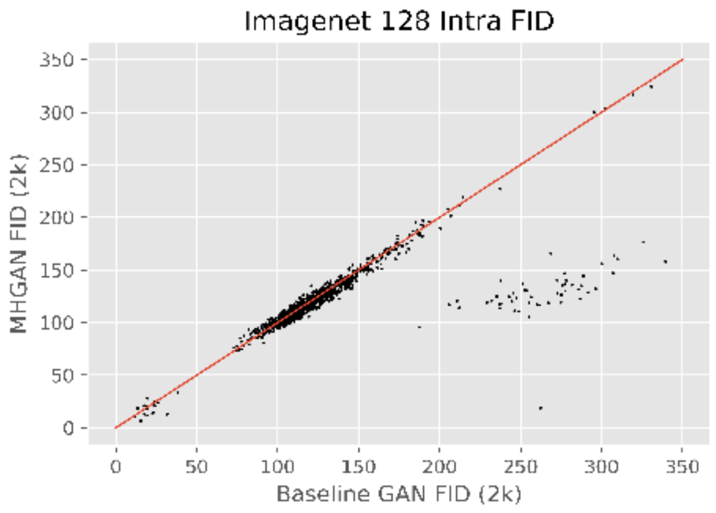
GANs sometimes explode



(a)



(b)



(c)

Actually: often

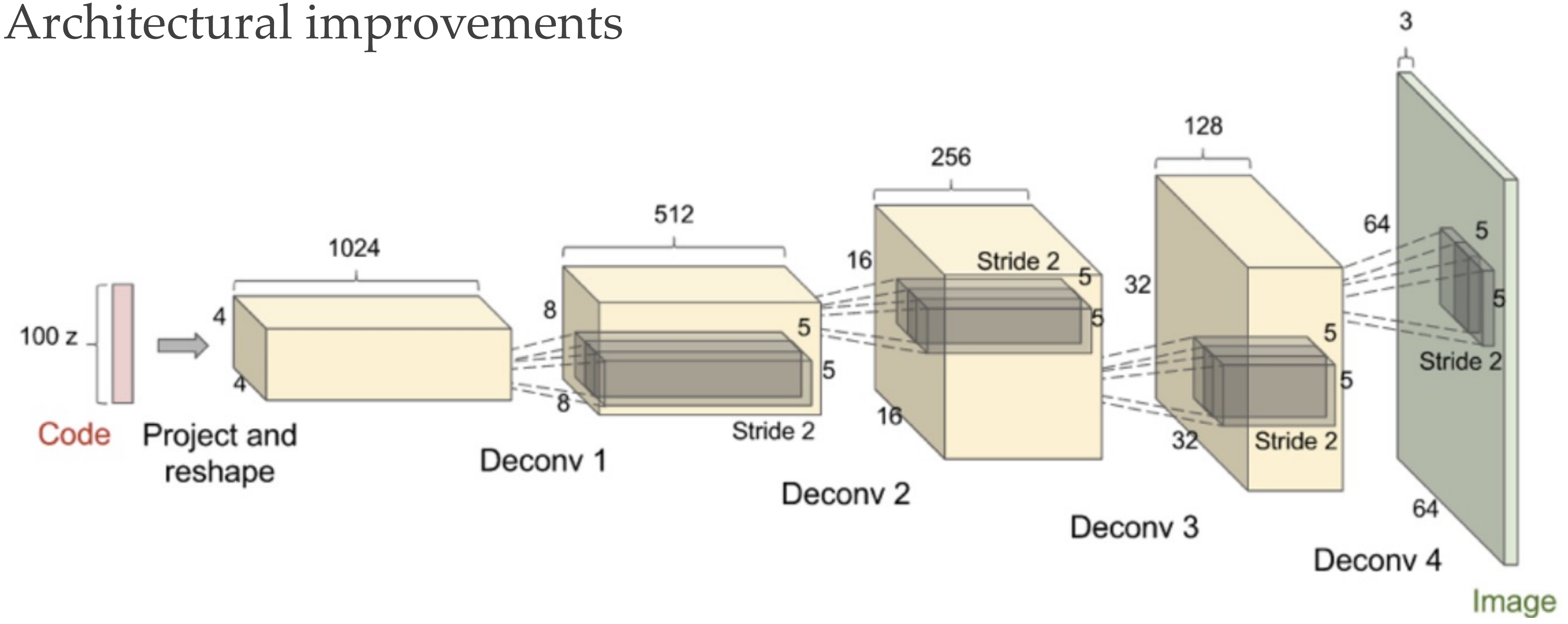
GAN models and applications

From -->

To -->

DCGAN

- One of the first scaling ups of GANs
  - Architectural improvements

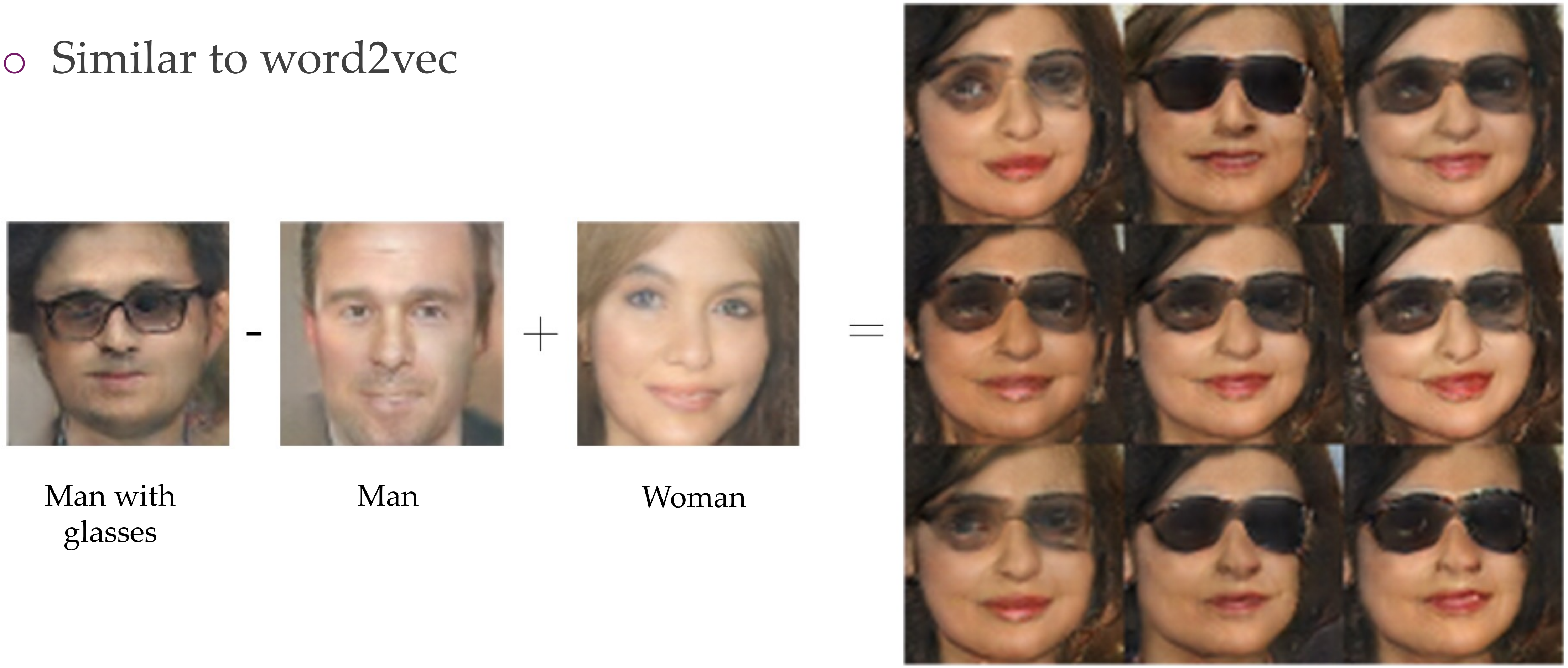


Radford, Metz, Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

Examples

Even vector space arithmetics …

- Similar to word2vec



Can generate new views

- Now GANs good enough to create fake data that can be used for training

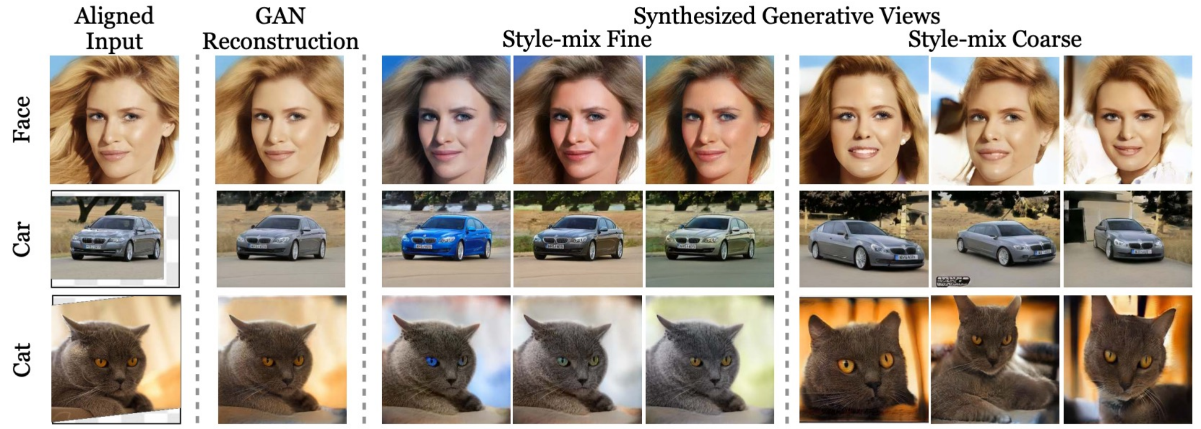


Figure 2: Synthesizing deep generative views. We first align (Aligned Input) and reconstruct an image by finding the corresponding latent code in StyleGAN2 [31] (GAN Reconstruction). We then investigate different approaches to produce image variations using the GAN, such as style-mixing on fine layers (Style-mix Fine), which predominantly changes color, or coarse layers (Style-mix Coarse), which changes pose. We show additional perturbations in supplementary material.

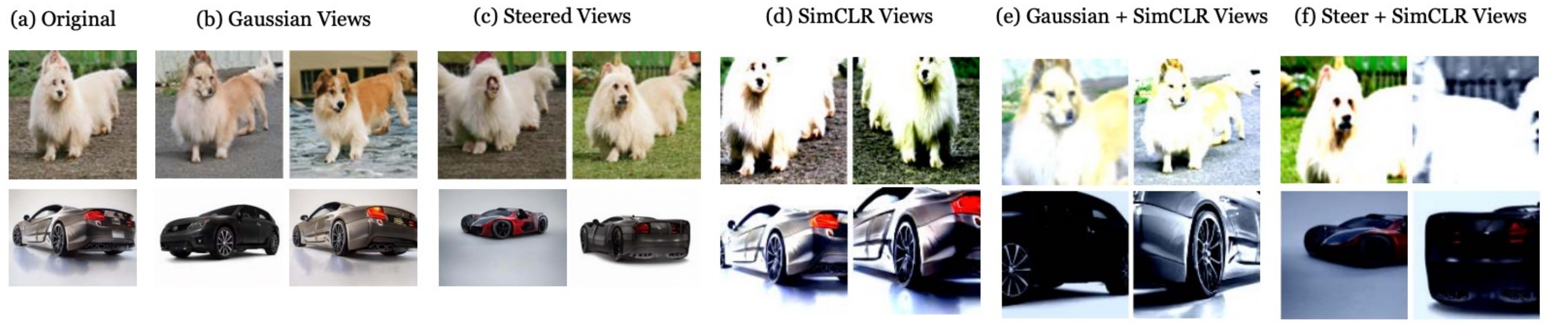


Figure 4: Examples of different transformation methods for unconditional IGM data. Top row shows samples of BigBiGAN trained on ImageNet1000, and the bottom row shows samples from the StyleGAN2 LSUN CAR.

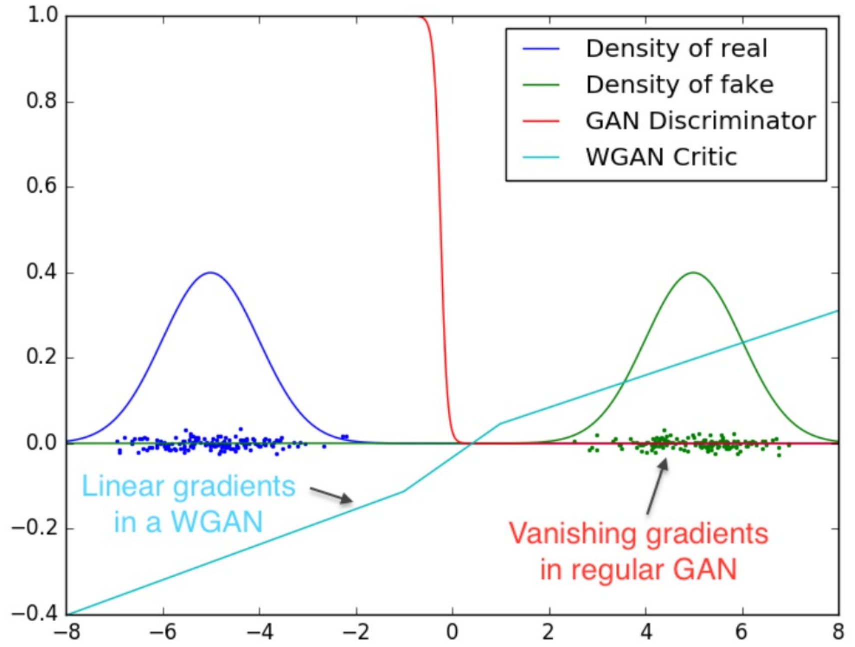
Generative models as a data source for multiview representation learning. Jahanian et al. ICLR 2022

Wasserstein GAN

- Instead of KL/JS, use Wasserstein (Earth Mover's) Distance

$$W\left(p_r, p_g\right) = \inf_{\gamma \sim \Pi(\mathrm{p_r}, \mathrm{p_g})} \mathrm{E}_{(\mathrm{x},\mathrm{y}) \sim \gamma} |x - y|$$

- Even for non-overlapping supports, the distance is meaningful



Arjovsky, Chintala, Bottou, Wasserstein GAN

Differences in GANs

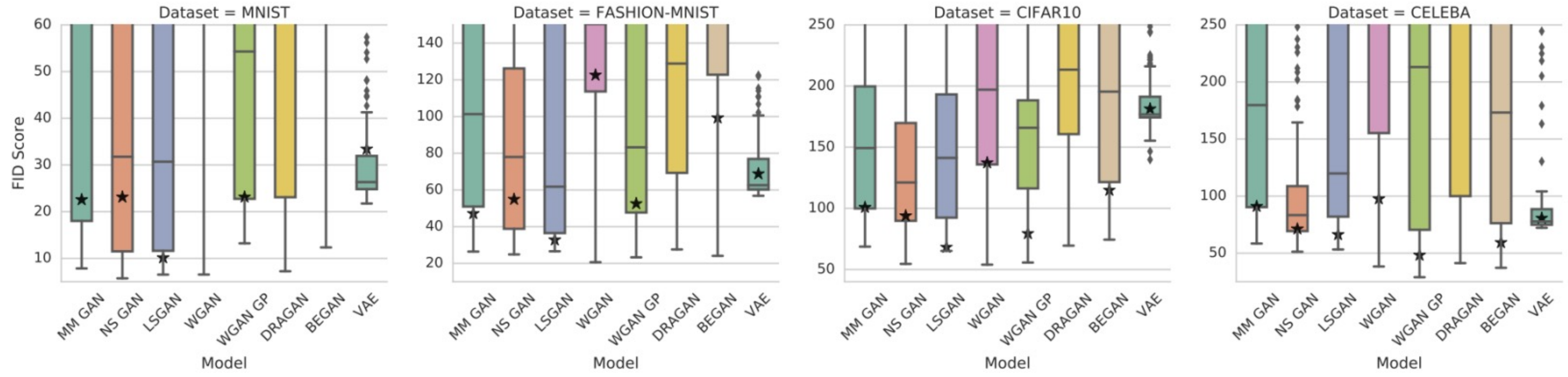
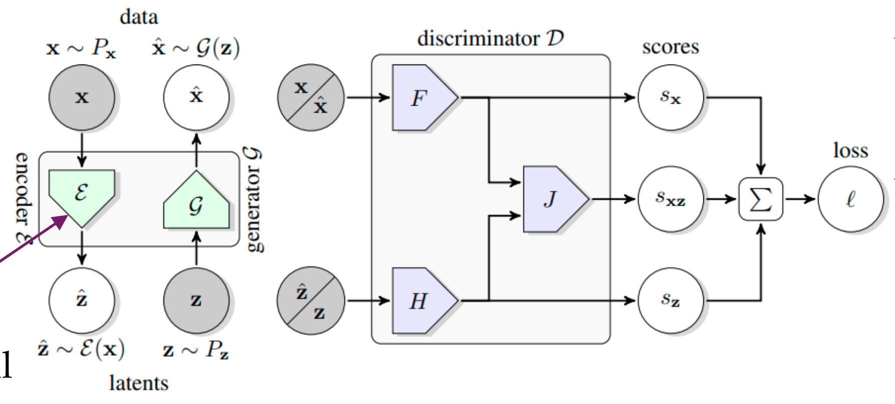


Figure 4: A *wide range* hyperparameter search (100 hyperparameter samples per model). Black stars indicate the performance of suggested hyperparameter settings. We observe that GAN training is extremely sensitive to hyperparameter settings and there is no model which is significantly more stable than others.

[“Are all GANs Created Equal?”, Lucic*, Kurach*, et al. 2018]

BigBiGAN

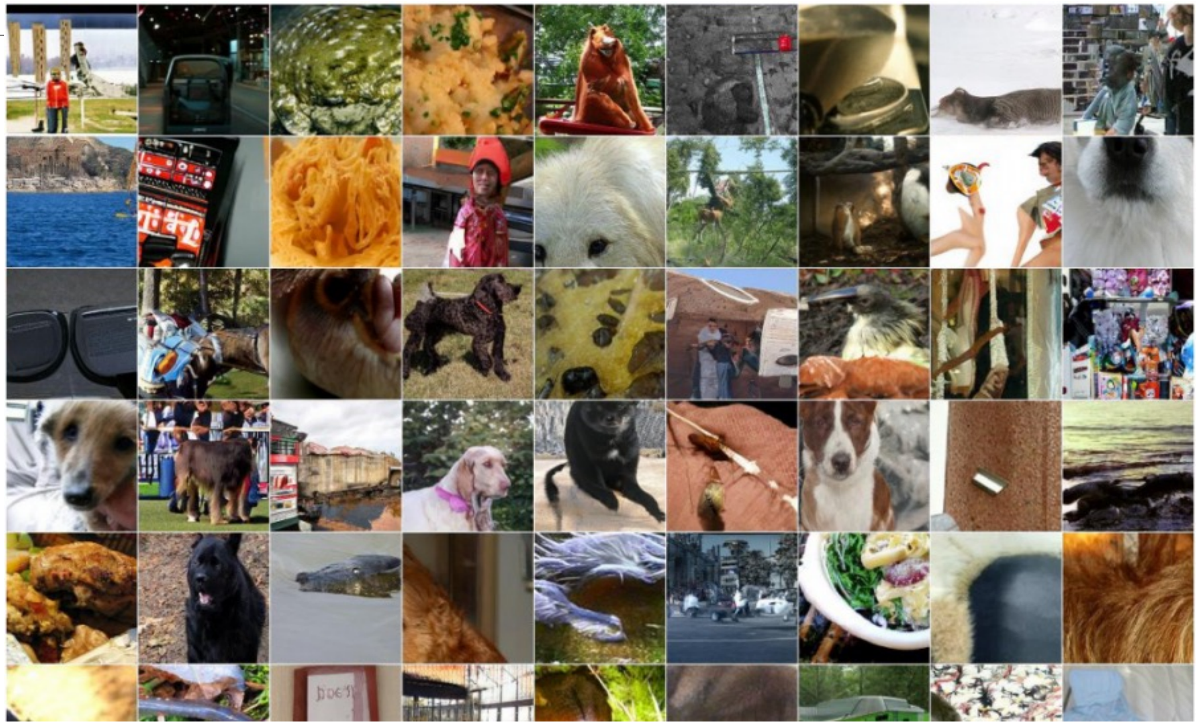


data
$\mathbf{x} \sim P_{\mathbf{x}}$
$\hat{\mathbf{x}} \sim \mathcal{G}(\mathbf{z})$
encoder $\mathcal{E}$
generator $\mathcal{G}$
$\hat{\mathbf{z}} \sim \mathcal{E}(\mathbf{x})$
$\mathbf{z} \sim P_{\mathbf{z}}$
latents
discriminator $\mathcal{D}$
F
J
H
scores
$s_{\mathbf{x}}$
$s_{\mathbf{xz}}$
$s_{\mathbf{z}}$
loss
ℓ

additional

- Discriminator works on (x,z)
- Therefore we need an encoder that maps real x to z: E(x)=z
- For the fake data, we just use the sampled z
- This Encoder E learns strong representations
- E is "part of" discriminator

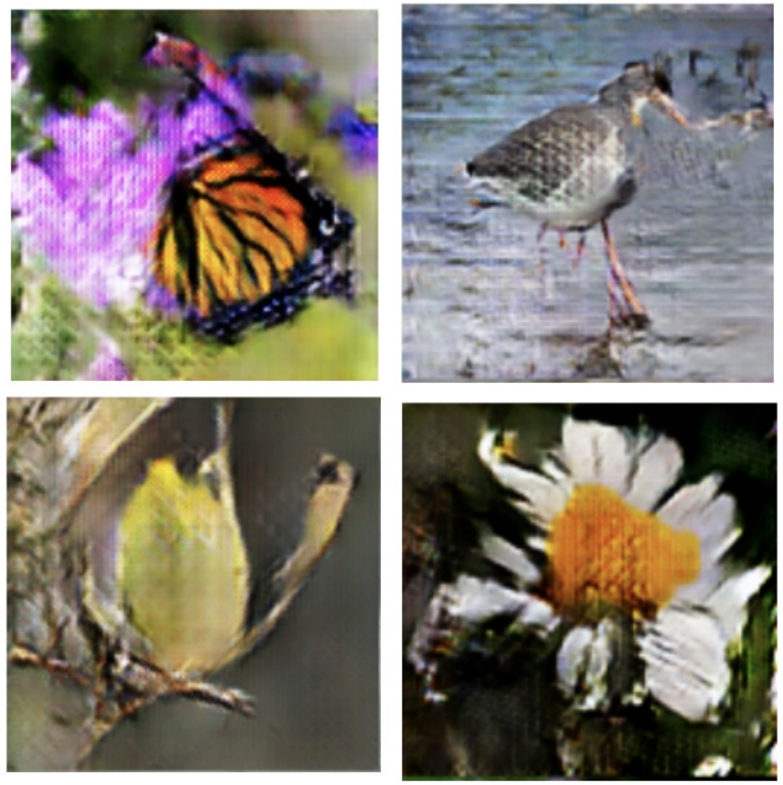
Generated images:



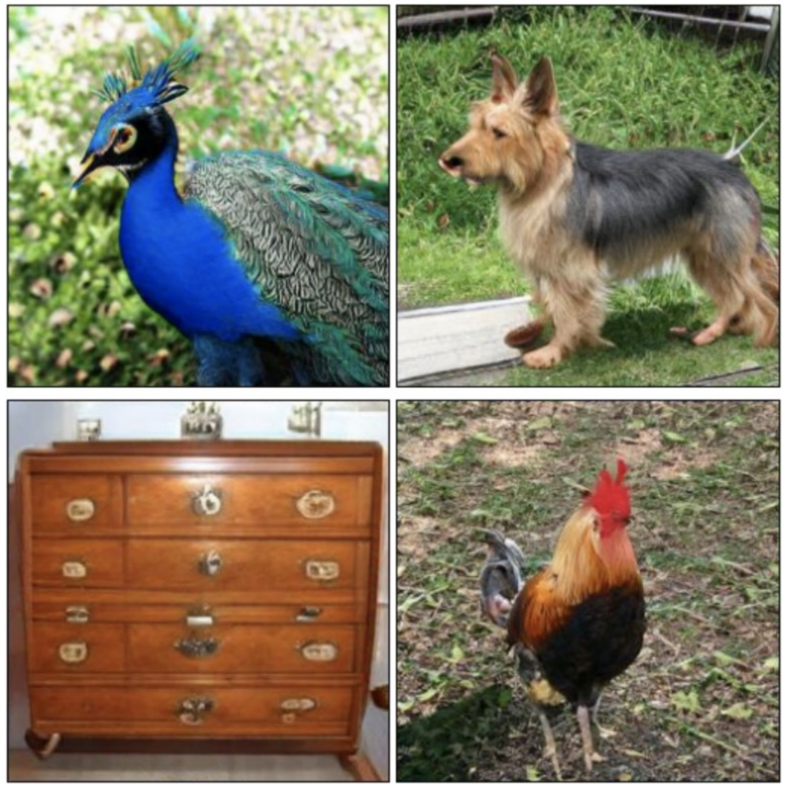
Large Scale Adversarial Representation Learning. Donahue et al. NeurIPS 2019

So what changed? More data? -- No

ACGAN [Odena et al. 2016]



BigGAN [Brock et al. 2018]

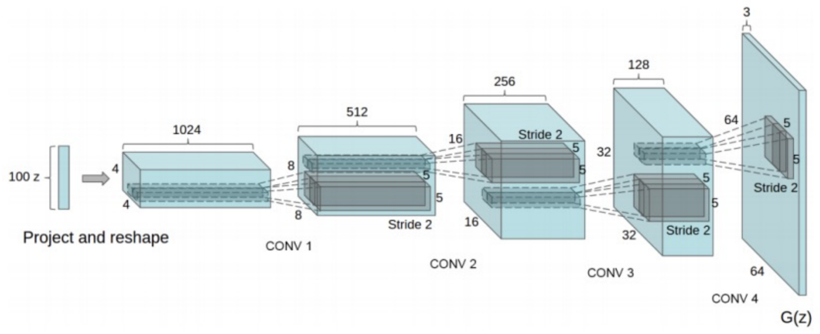


Both trained on Imagenet

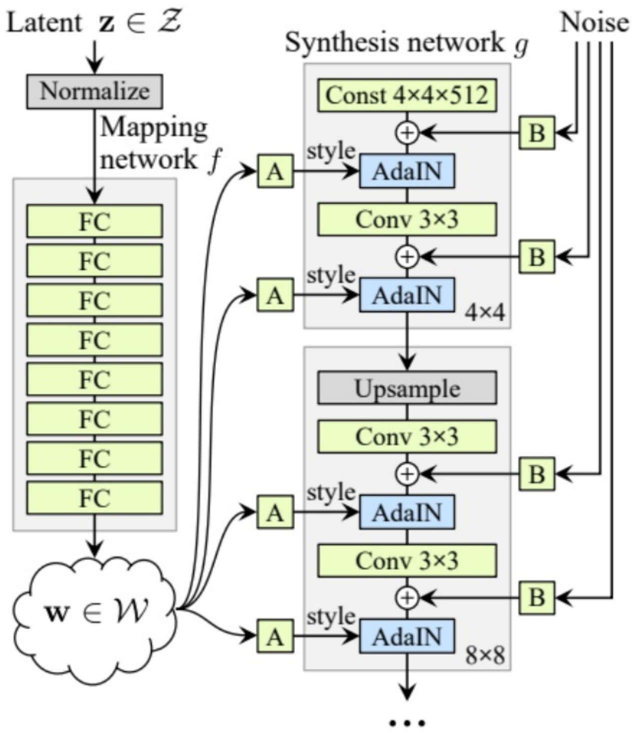
So what changed? Architectures and compute: yes

Architectures

DCGAN
[Radford, Metz, Chintala 2016]



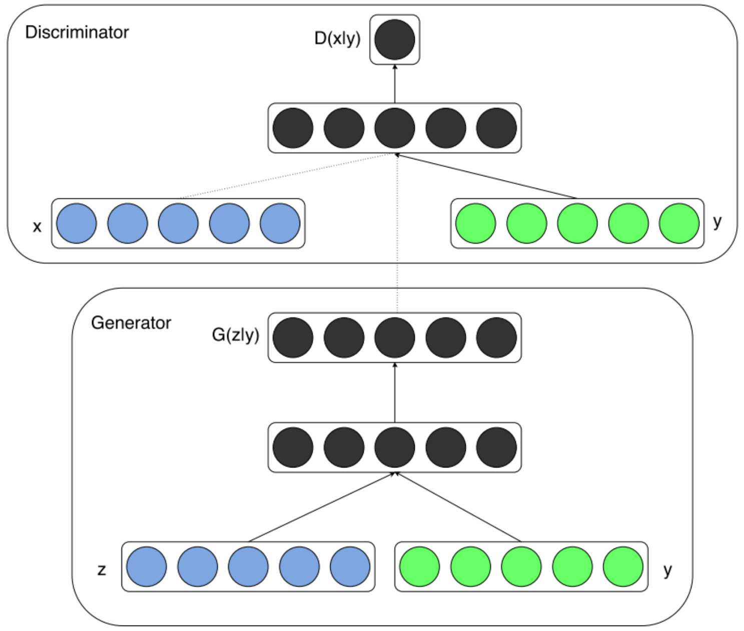
StyleGAN
[Karras, Laine, Aila 2019]



Conditional GAN

- Conditioning on labels
  - Appending label/annotation vector to noise vector

$$\min_{G}\max_{D}\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x}|\boldsymbol{y})]+\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z}|\boldsymbol{y})))]$$



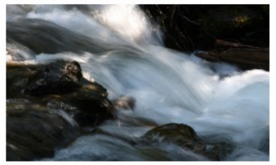
	User tags + annotations	Generated tags
	montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails
	food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
	water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
	people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life

Table 2: Samples of generated tags

Mirza and Osindero, Conditional Generative Adversarial Nets

Image to image translation

- Conditioning GAN on other images (like edges)

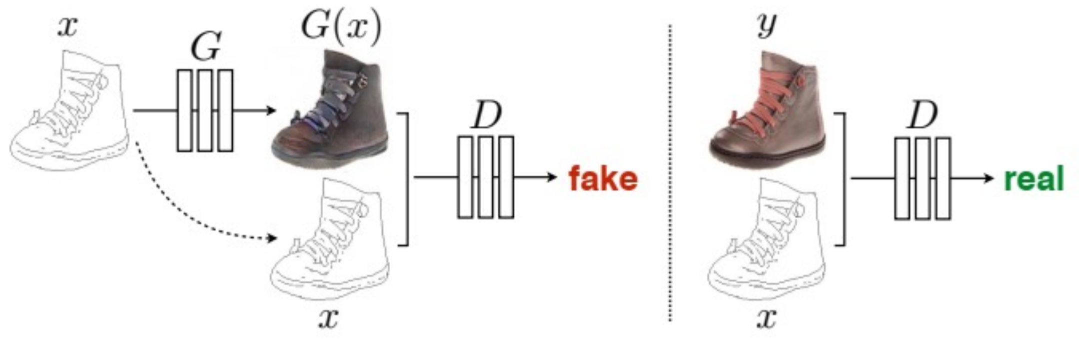


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

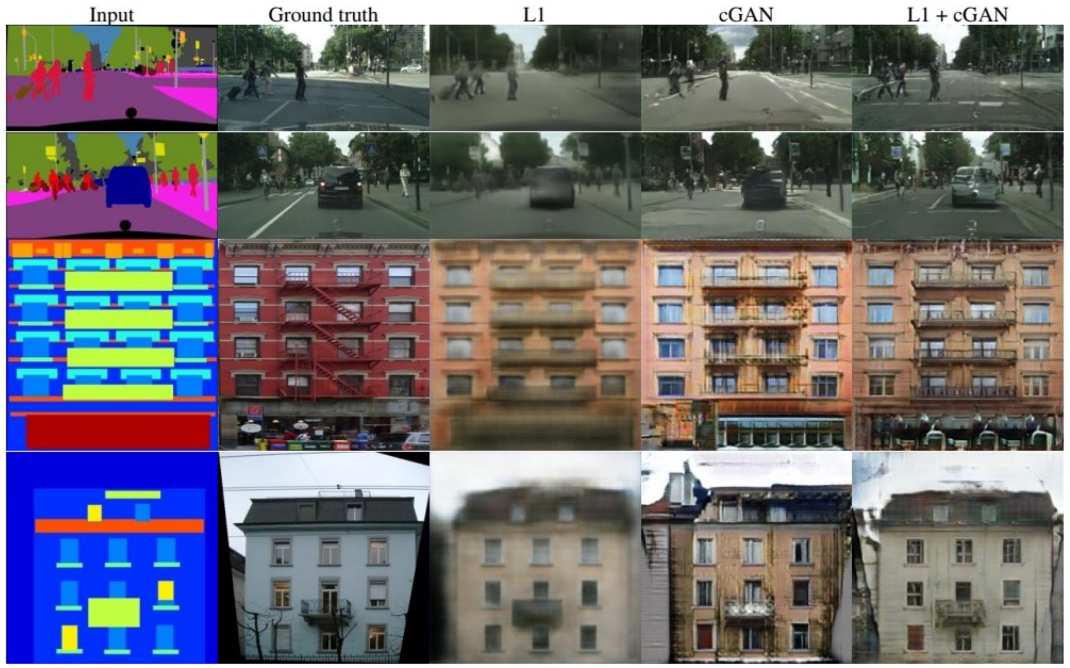


Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

Isola, Zhu, Zhou, Efros, Image-to-Image Translation with Conditional Adversarial Networks

Adversarial AutoEncoders: and adversarial network in latent space

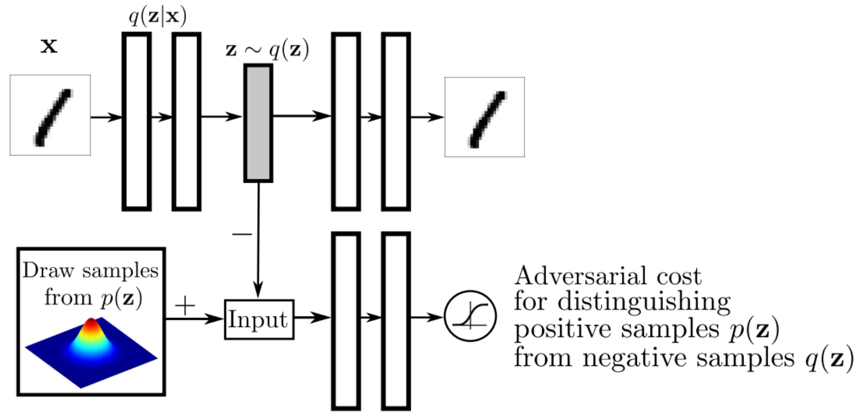

x
$q(\mathbf{z}|\mathbf{x})$
$\mathbf{z} \sim q(\mathbf{z})$
Draw samples from $p(\mathbf{z})$
+
−
Input
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

Figure 1: Architecture of an adversarial autoencoder. The top row is a standard autoencoder that reconstructs an image $\mathbf{x}$ from a latent code $\mathbf{z}$. The bottom row diagrams a second network trained to discriminatively predict whether a sample arises from the hidden code of the autoencoder or from a sampled distribution specified by the user.

- q(z|x)
  - Deterministic
  - Gaussian (via reparametrization)
- Compared to VAE:
  - VAE uses KL to give p(z|x) structure
  - AAE uses an adversarial network for this
  - Can toggle between AE and AAE
- Can be extended for conditional modelling

Adversarial Autoencoders. Makhzani et al. 2016

CycleGAN: "img2img" models

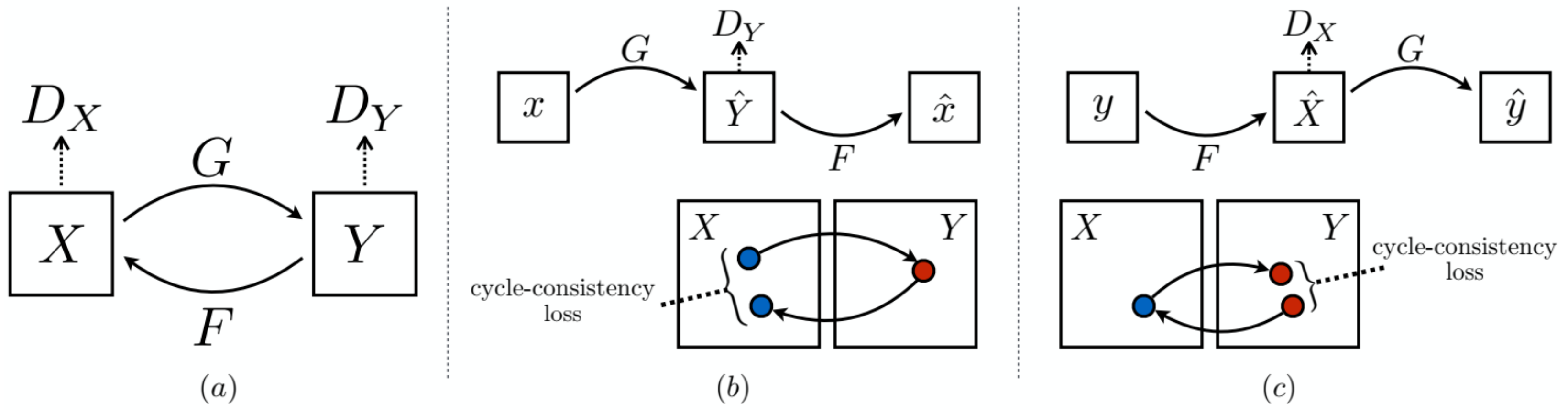
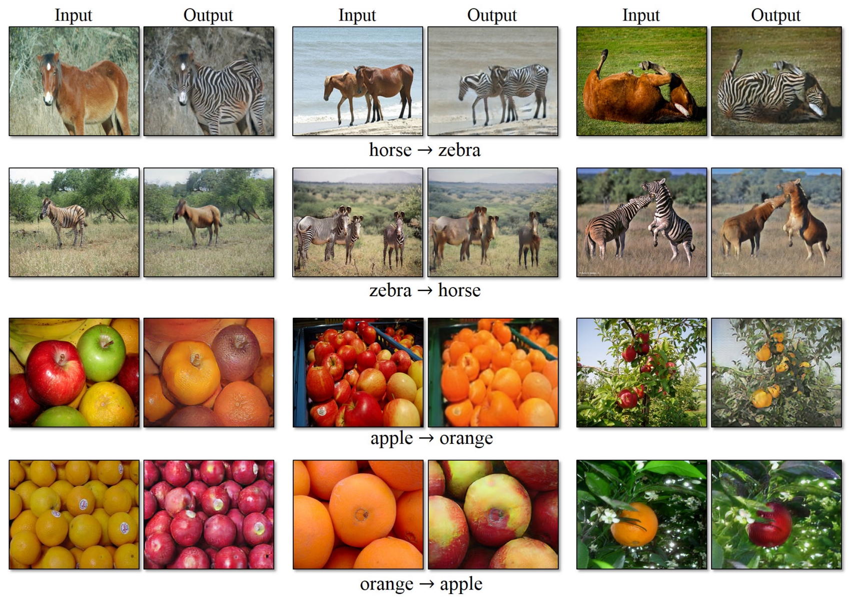


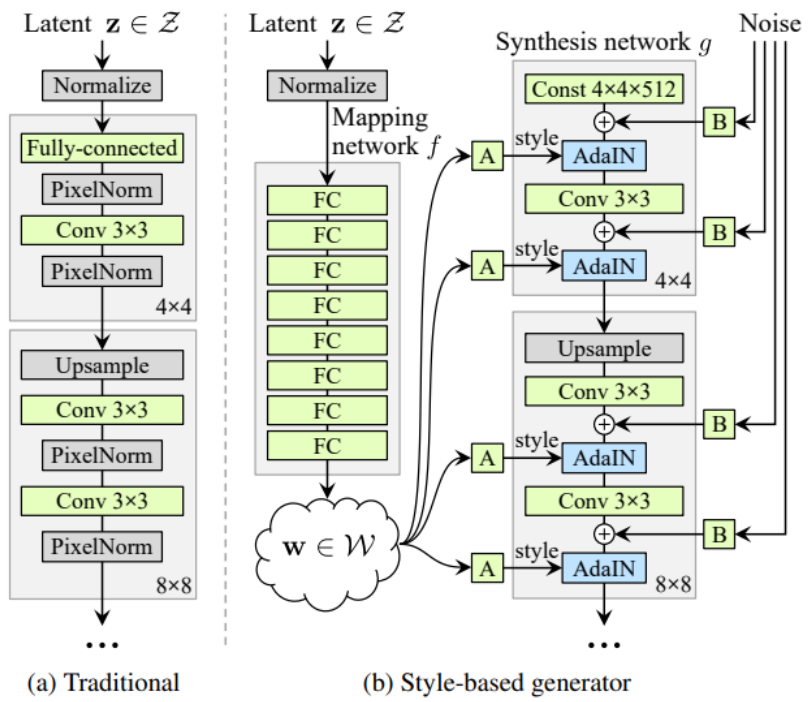
Figure 3: (a) Our model contains two mapping functions $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \to G(x) \to F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \to F(y) \to G(F(y)) \approx y$

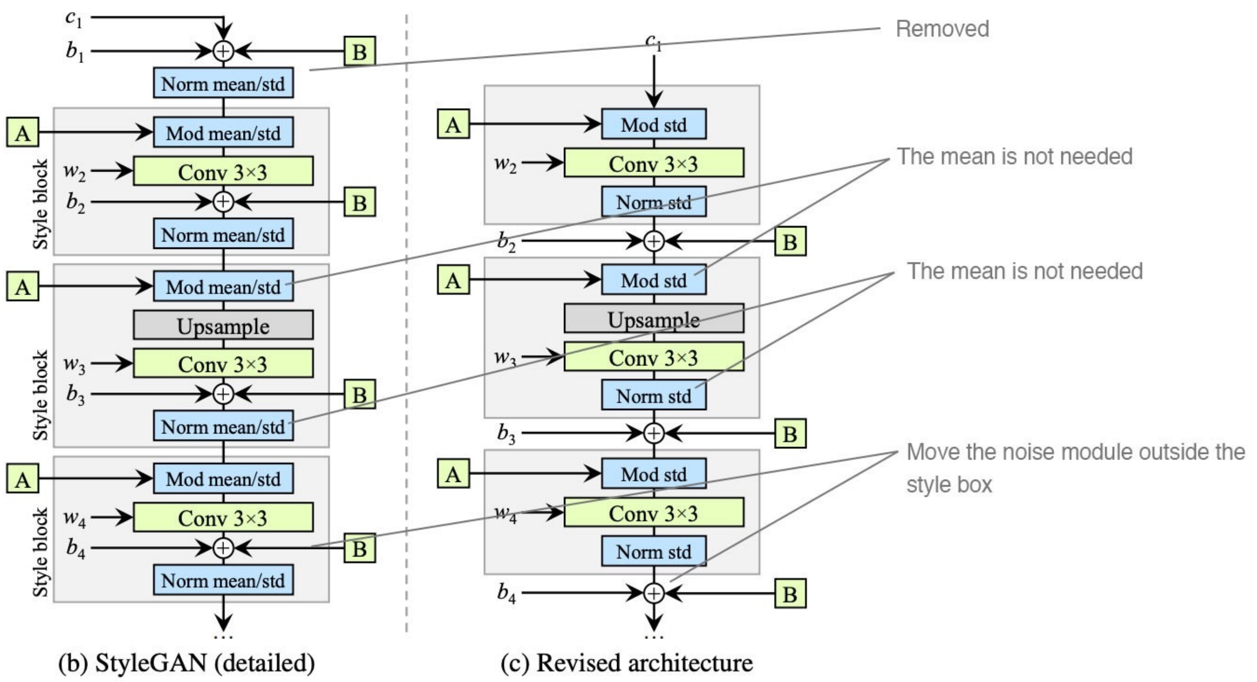


Zhu, Park, Isola, Efros, Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

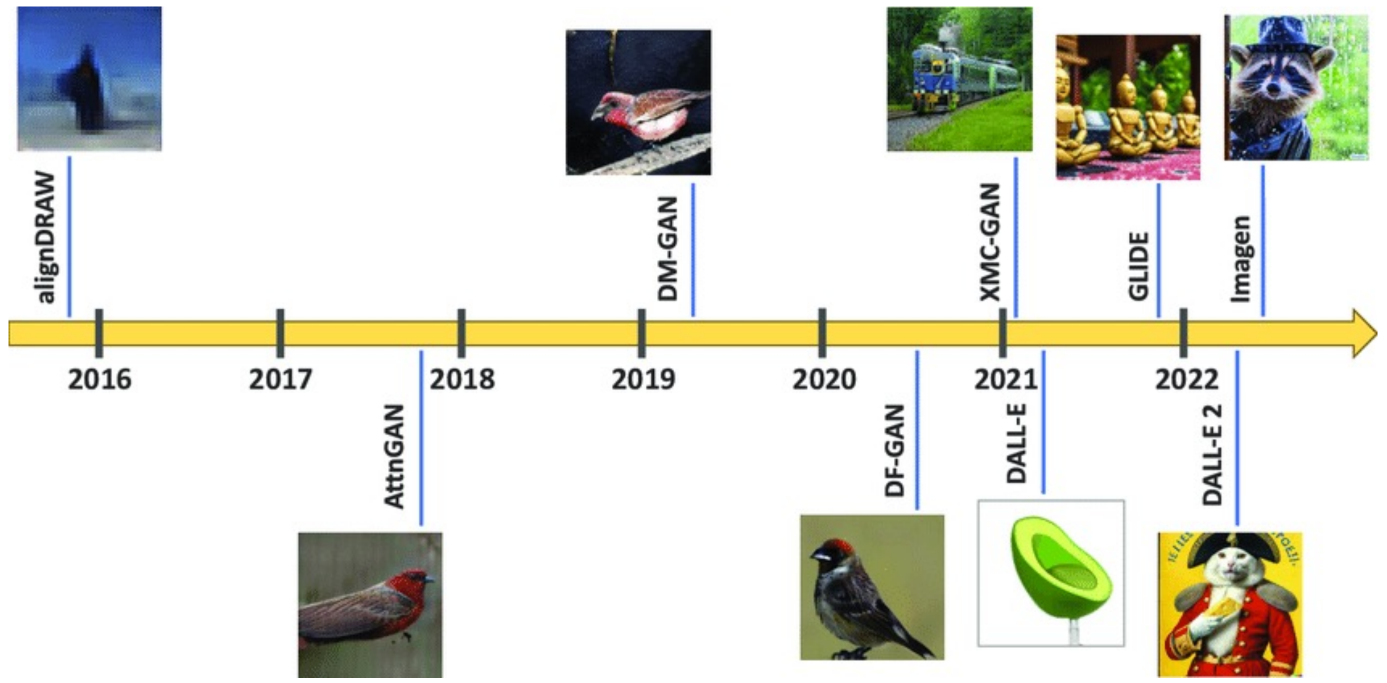
StyleGAN and StyleGANv2

- Architectural innovations (tinkering)
- Scaling





Text-image generative models



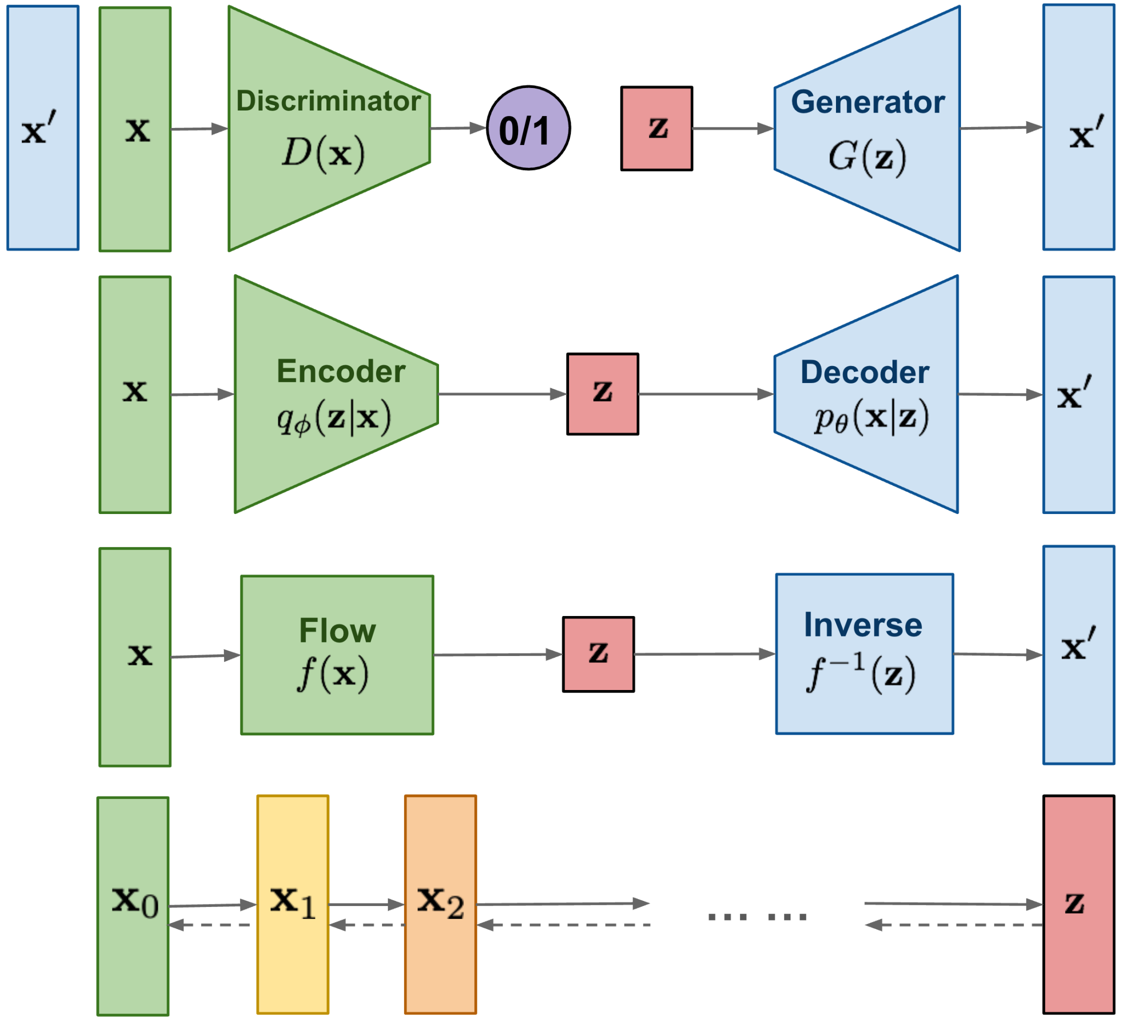
Overview of methods

GAN: Adversarial training

VAE: maximize variational lower bound

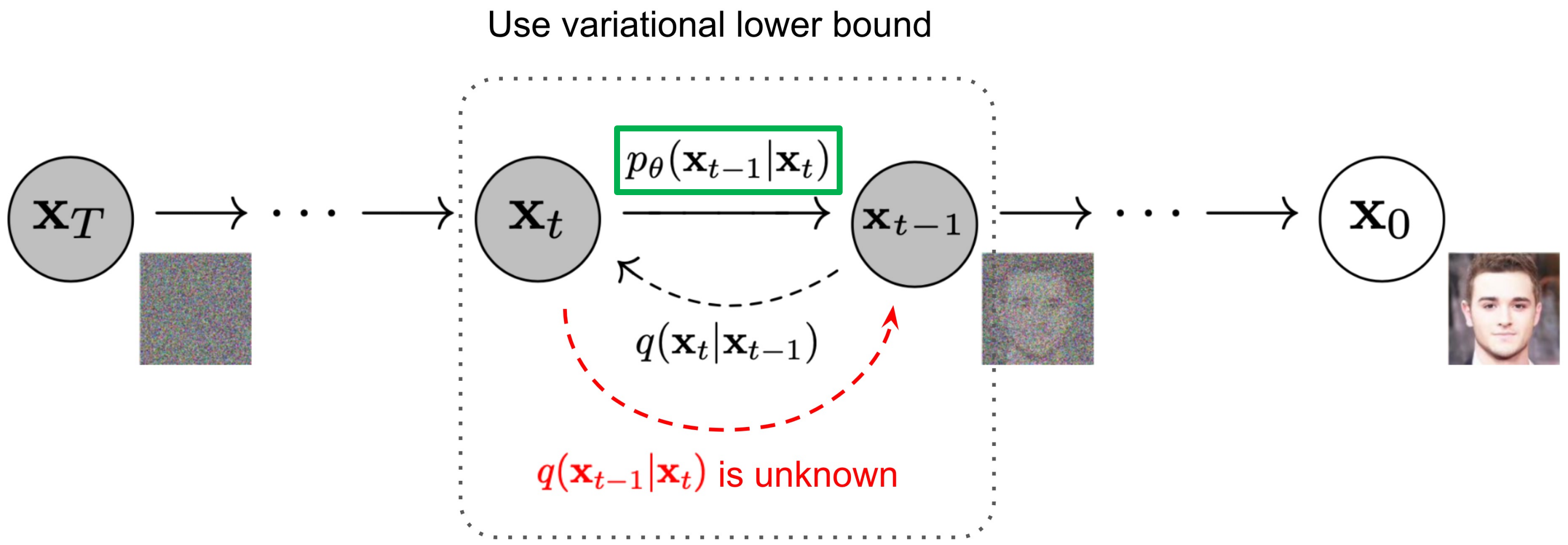
Flow-based models: Invertible transform of distributions

Diffusion models: Gradually add Gaussian noise and then reverse



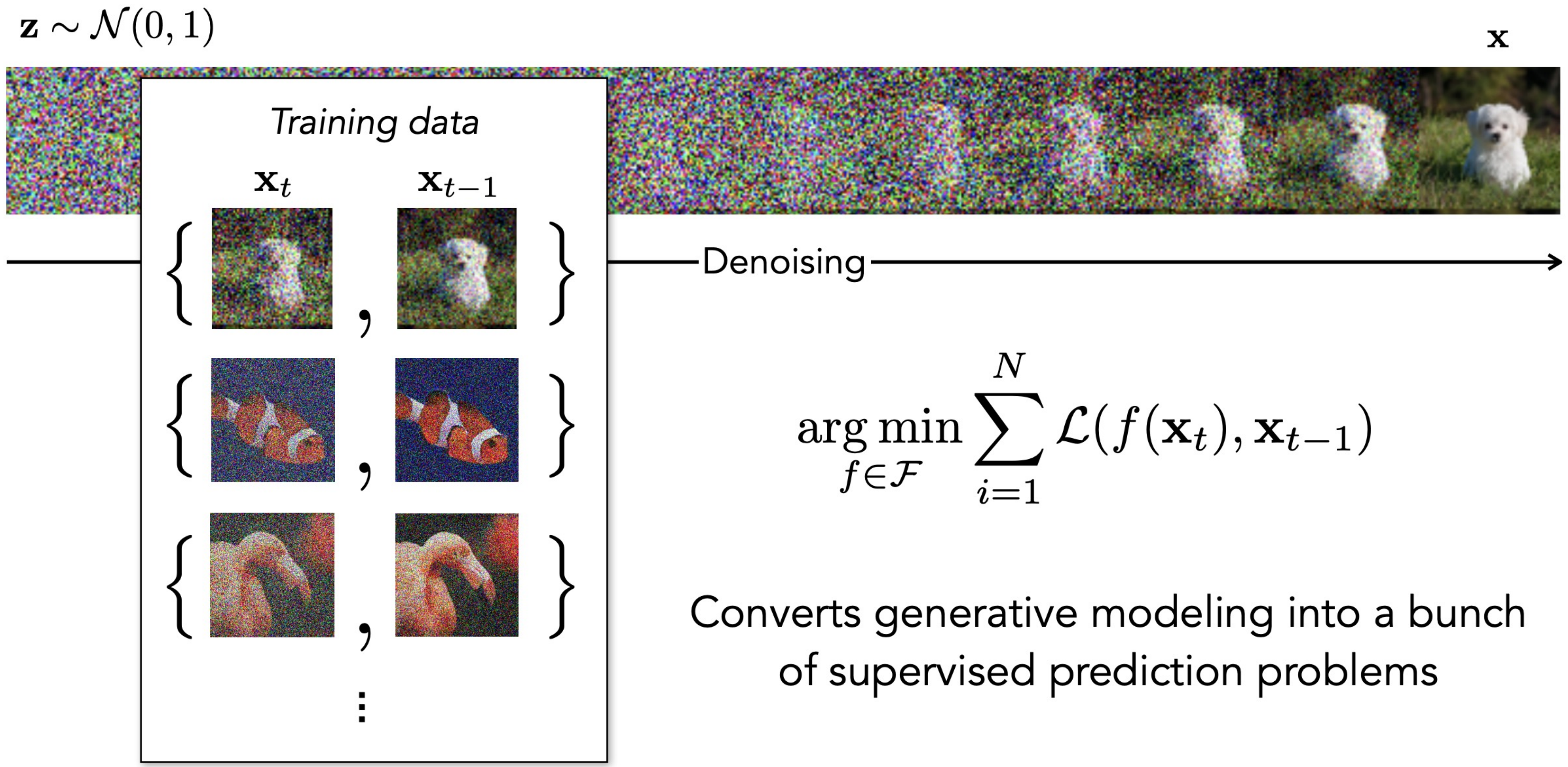
Basic idea of diffusion models: learning how to denoise

- Imagine we gradually add noise to an image, until it's Gaussian normally distributed.
- Now if we could simply learn how to reverse the process, we could generate images
  - By starting from random noise
- + nice maths (diffusion models are explicit density models)



Use variational lower bound
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown

Diffusion models turn generative learning into a sequence of supervised problems

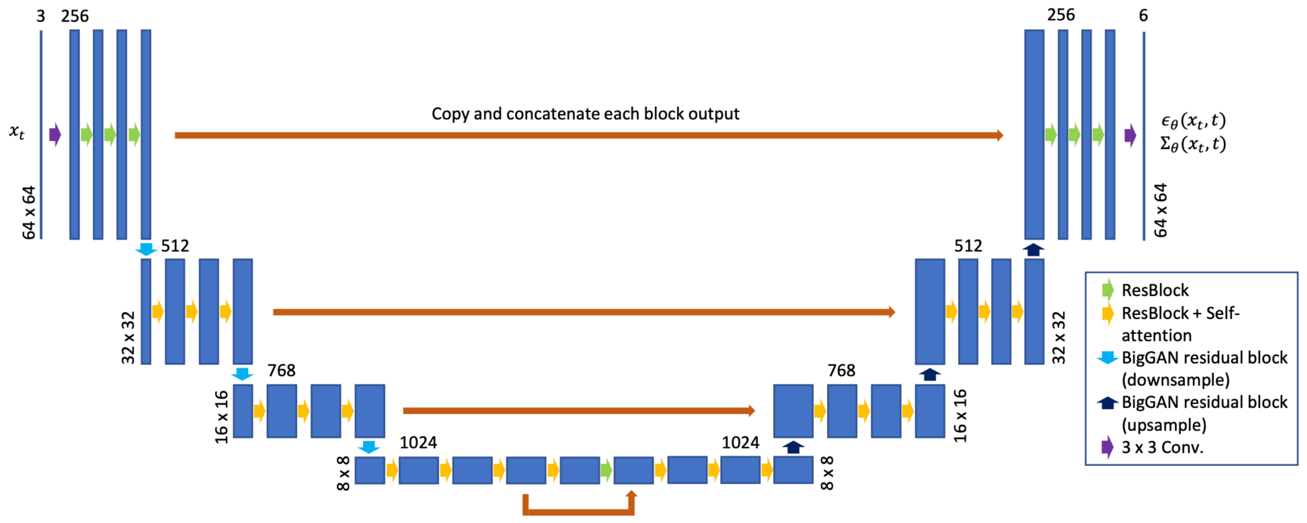


$$\arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_t), \mathbf{x}_{t-1})$$

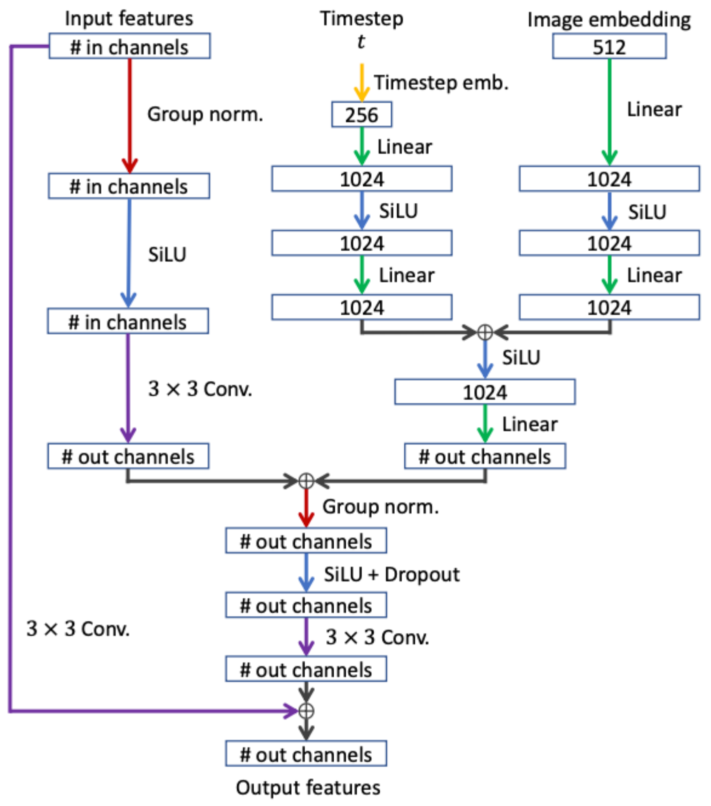
Converts generative modeling into a bunch of supervised prediction problems

The architecture: a modified U-Net that uses diffusion time t

- Denoising via U-Net that also has time as input



3
256
256
6
x_t
64 x 64
Copy and concatenate each block output
$\epsilon_\theta(x_t, t)$
$\Sigma_\theta(x_t, t)$
512
512
32 x 32
768
768
16 x 16
1024
1024
8 x 8
ResBlock
ResBlock + Self-attention
BigGAN residual block (downsample)
BigGAN residual block (upsample)
3 x 3 Conv.



Combining this with text as cond. inputs: DALL-E v2 / “unCLIP“

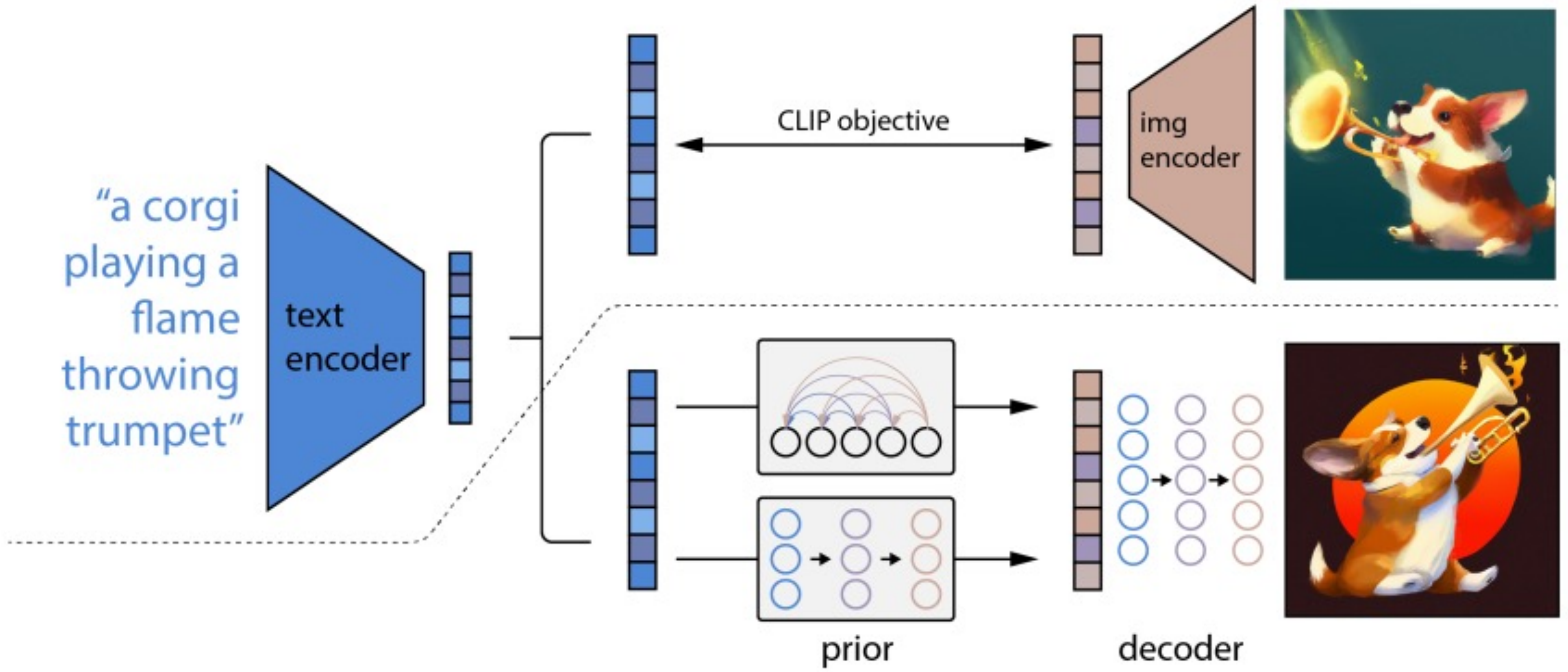
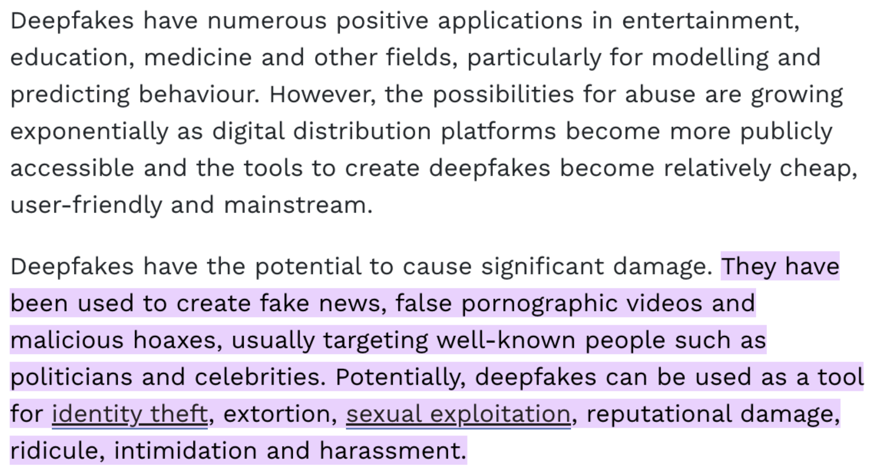


Figure 2: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

+ Don’t forget about the 400M sizes training dataset and the 3B parameters

Final note about deep fakes and ethics

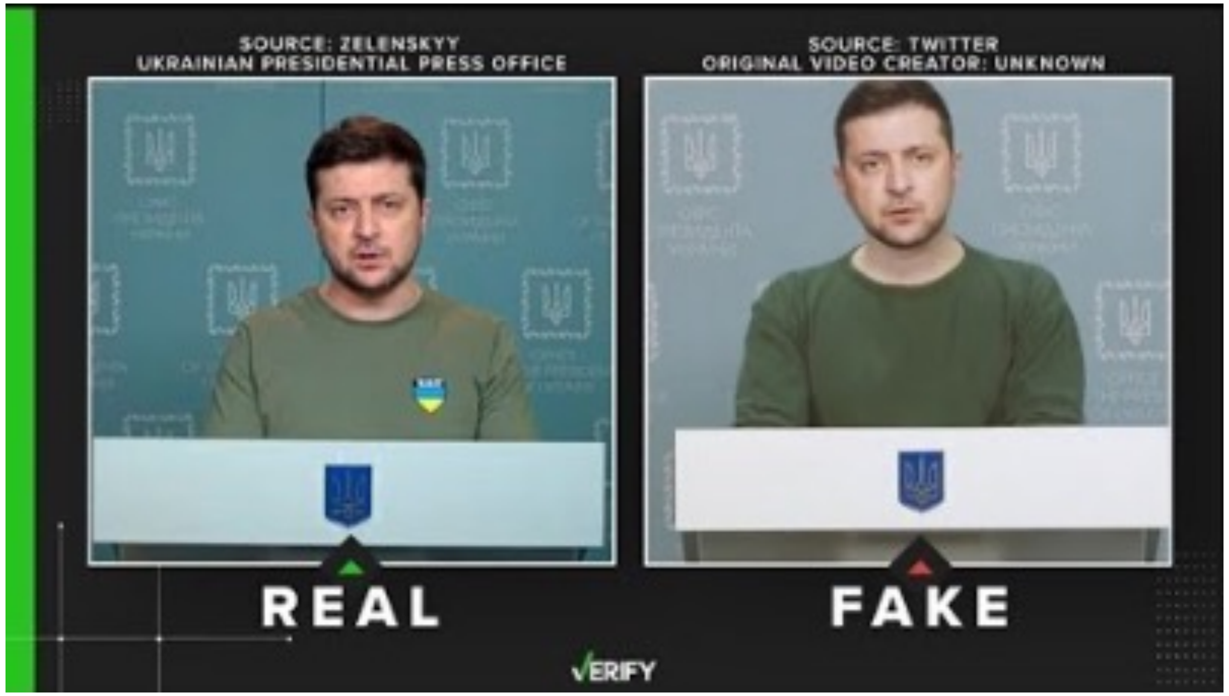
- Pay attention in the FACT-AI course! ☺
- Deep fakes are a problem



Deepfakes have numerous positive applications in entertainment, education, medicine and other fields, particularly for modelling and predicting behaviour. However, the possibilities for abuse are growing exponentially as digital distribution platforms become more publicly accessible and the tools to create deepfakes become relatively cheap, user-friendly and mainstream.

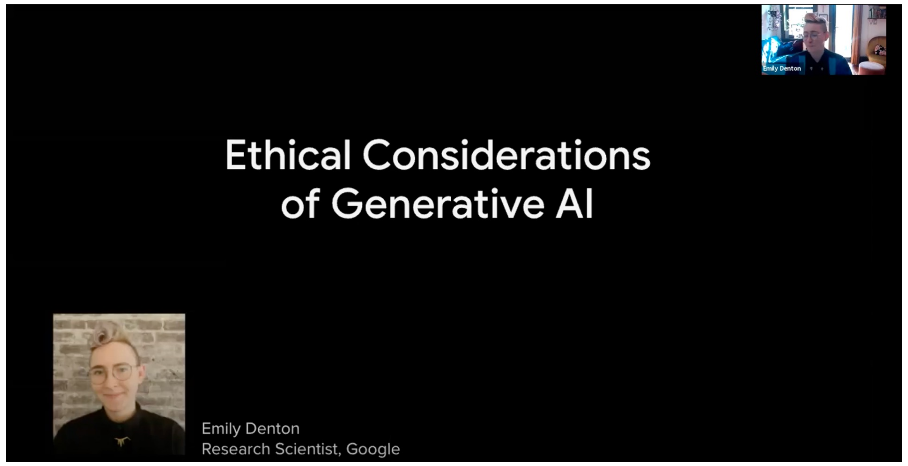
Deepfakes have the potential to cause significant damage. They have been used to create fake news, false pornographic videos and malicious hoaxes, usually targeting well-known people such as politicians and celebrities. Potentially, deepfakes can be used as a tool for identity theft, extortion, sexual exploitation, reputational damage, ridicule, intimidation and harassment.

- Arms-race between fakers and detectors.


SOURCE: ZELENSKYY
UKRAINIAN PRESIDENTIAL PRESS OFFICE
SOURCE: TWITTER
ORIGINAL VIDEO CREATOR: UNKNOWN
REAL
FAKE
VERIFY

https://www.esafety.gov.au/industry/tech-trends-and-challenges/deepfakes

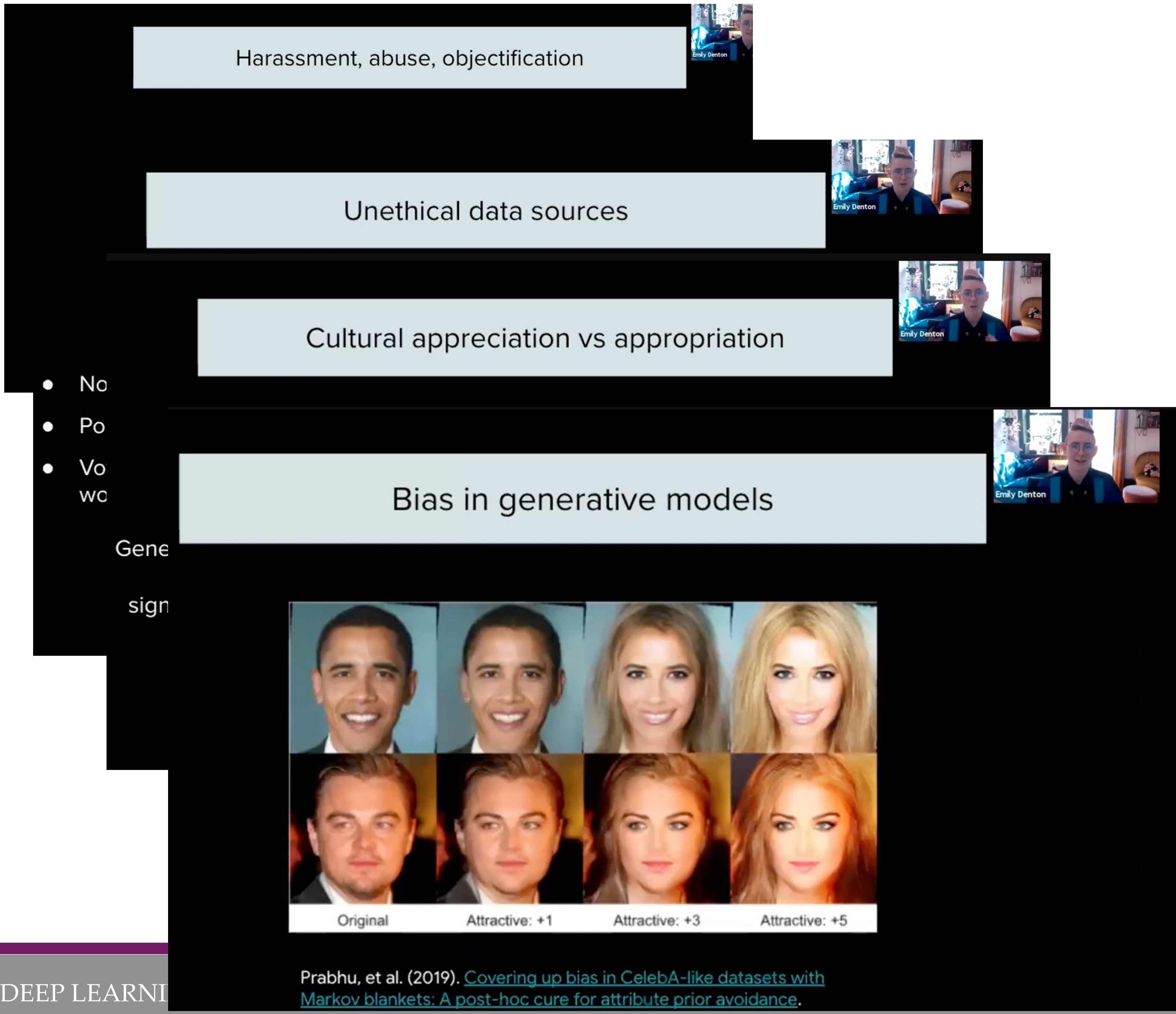
Recommended watch (just 34min)



Ethical Considerations of Generative AI

Emily Denton
Research Scientist, Google

- [Emily Denton on Ethical Considerations of Generative AI]



Harassment, abuse, objectification

Unethical data sources

Cultural appreciation vs appropriation

- No
- Po
- Vo wo

Gene

sign

Bias in generative models

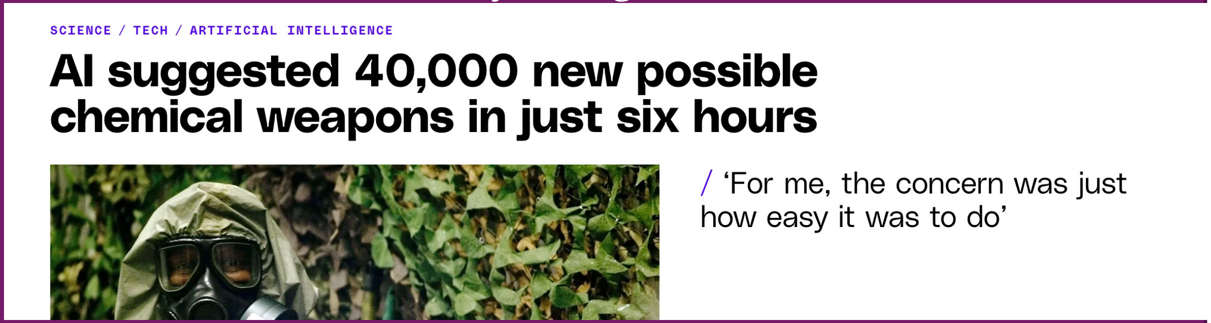
Prabhu, et al. (2019). Covering up bias in CelebA-like datasets with Markov blankets: A post-hoc cure for attribute prior avoidance.

Quiz: what dimensions need to be considered when thinking about developing the next generative model?

1) Ethics of dataset used (bias, representation, consent)
2) Copyright of inputs and copyright of outputs
3) Malicious usecases of the trained model
4) Malicious usecases of adapting the trained model/method
5) Effect on surveillance/creative industry/art/fake news
6) All of the above and more

See also "Broader Impacts" section in papers, e.g. this guide

Not only images

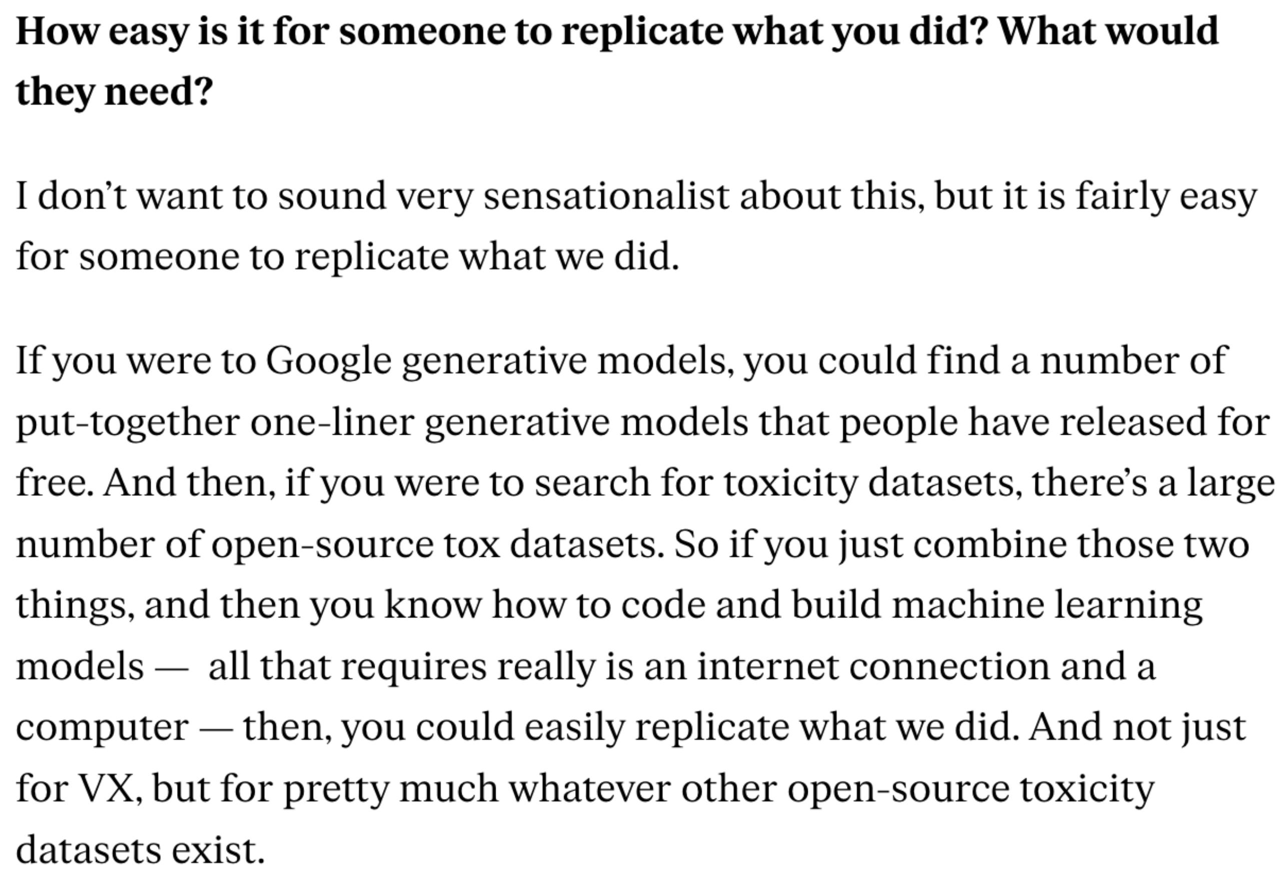

SCIENCE / TECH / ARTIFICIAL INTELLIGENCE

AI suggested 40,000 new possible chemical weapons in just six hours

/ 'For me, the concern was just how easy it was to do'

More on that previous one..

- https://www.theverge.com/2022/3/17/22983197/ai-new-possible-chemical-weapons-generative-models-vx



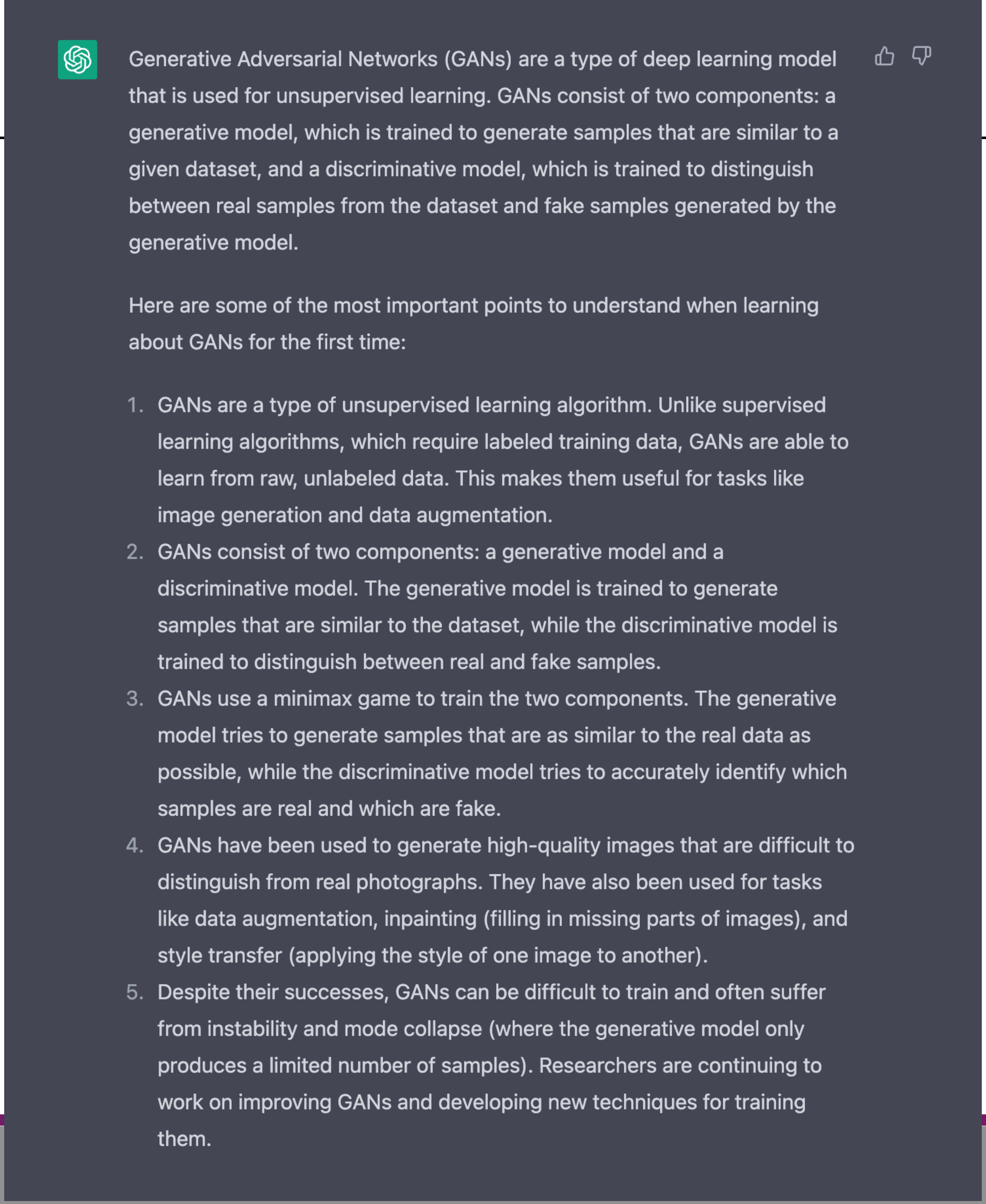
How easy is it for someone to replicate what you did? What would they need?

I don't want to sound very sensationalist about this, but it is fairly easy for someone to replicate what we did.

If you were to Google generative models, you could find a number of put-together one-liner generative models that people have released for free. And then, if you were to search for toxicity datasets, there's a large number of open-source tox datasets. So if you just combine those two things, and then you know how to code and build machine learning models — all that requires really is an internet connection and a computer — then, you could easily replicate what we did. And not just for VX, but for pretty much whatever other open-source toxicity datasets exist.

Summary of GANs

- Actually have nothing to add
- https://chat.openai.com/chat



Generative Adversarial Networks (GANs) are a type of deep learning model that is used for unsupervised learning. GANs consist of two components: a generative model, which is trained to generate samples that are similar to a given dataset, and a discriminative model, which is trained to distinguish between real samples from the dataset and fake samples generated by the generative model.

Here are some of the most important points to understand when learning about GANs for the first time:

1. GANs are a type of unsupervised learning algorithm. Unlike supervised learning algorithms, which require labeled training data, GANs are able to learn from raw, unlabeled data. This makes them useful for tasks like image generation and data augmentation.
2. GANs consist of two components: a generative model and a discriminative model. The generative model is trained to generate samples that are similar to the dataset, while the discriminative model is trained to distinguish between real and fake samples.
3. GANs use a minimax game to train the two components. The generative model tries to generate samples that are as similar to the real data as possible, while the discriminative model tries to accurately identify which samples are real and which are fake.
4. GANs have been used to generate high-quality images that are difficult to distinguish from real photographs. They have also been used for tasks like data augmentation, inpainting (filling in missing parts of images), and style transfer (applying the style of one image to another).
5. Despite their successes, GANs can be difficult to train and often suffer from instability and mode collapse (where the generative model only produces a limited number of samples). Researchers are continuing to work on improving GANs and developing new techniques for training them.

Summary

- Implicit density models: Motivation
- Generative adversarial networks
- Challenges
- GAN models

Reading materials:

- Book [4]: Chapter 10.4
- Papers mentioned in the slides