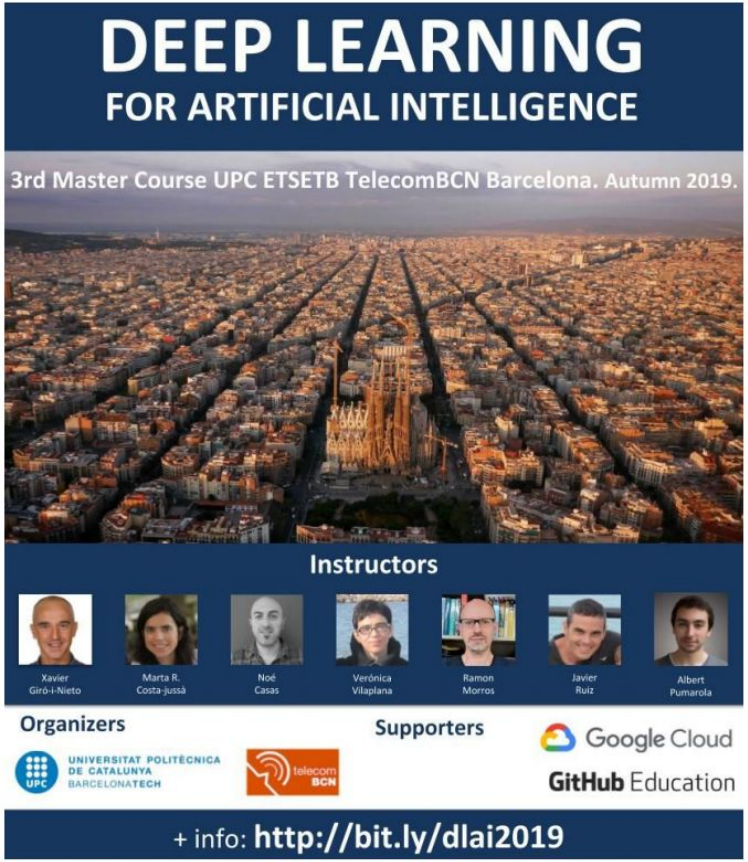


#DLBCN

Generative Adversarial Networks (GANs)

Albert Pumarola
apumarola@iri.upc.edu

https://telecombcn-dl.github.io/dlai-2019/

Acknowledgements

Previous Year Lecture

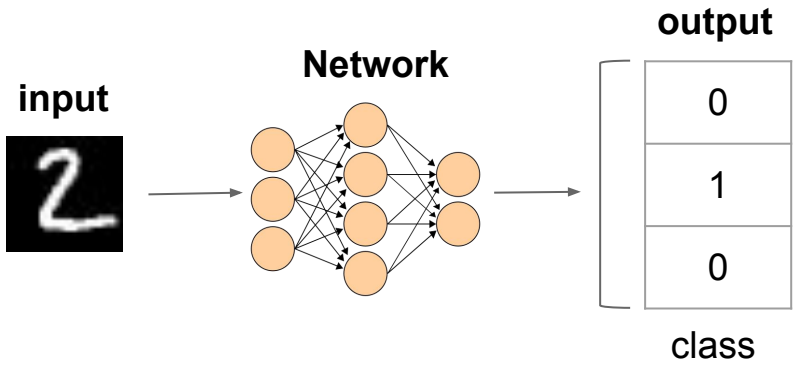
Today

- We are going to make a shift in the modeling paradigm: **discriminative → generative**.
- Is this a type of neural network? No → either fully connected networks, convolutional networks or recurrent networks fit, depending on how we condition data points (sequentially, globally, etc.).
- This is a very fun topic with which we can make a network paint, sing or write.

Introduction

What we are used to do with Neural Nets

Discriminative model ➜ aka. tell me the probability of some ‘Y’ responses given ‘X’ inputs. Here we don’t care about the process that generated ‘X’. we just detect some patterns in the input to give an answer. This gives us some outcomes probabilities given some ‘X’ data: P(Y | X).



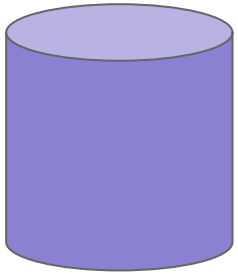
$$P(Y = [0,1,0] \mid X = [pixel_1, pixel_2, \ldots, pixel_{784}])$$

Figure credit: Javier Ruiz

What is a generative model?

We have datapoints that emerge from some generating process (landscapes in the nature, speech from people, etc.)

$\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$



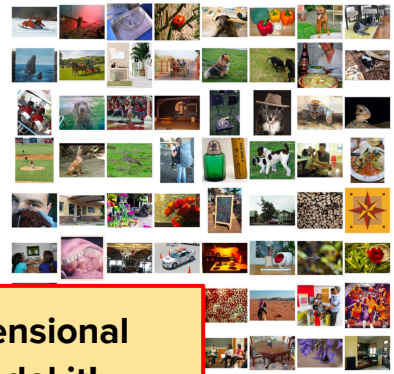
Having our dataset **X** with example datapoints (images, waveforms, written text, etc.) each point $\mathbf{x_n}$ lays in some M-dimensional space.

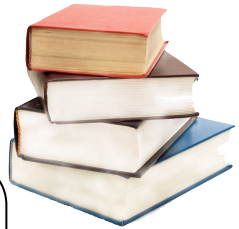
What is a generative model?

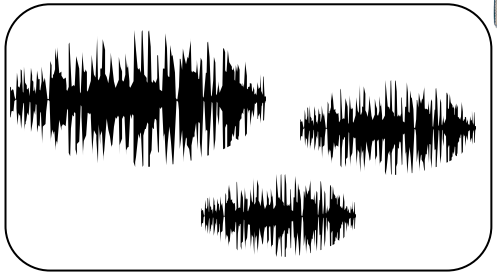
We have datapoints that emerge from some generating process (landscapes in the nature, speech from people, etc.)

$\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$

Each point x_n comes from an M-dimensional probability distribution $P(X)$ → Model it!







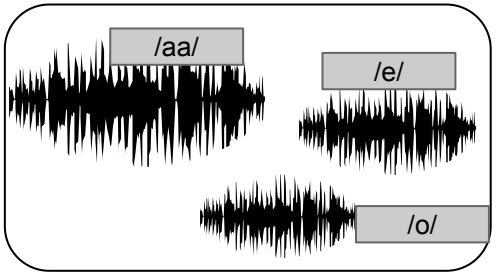
Having our dataset **X** with example datapoints (images, waveforms, written text, etc.) each point $\mathbf{x_n}$ lays in some M-dimensional space.

What is a generative model?

We have datapoints that emerge from some generating process (landscapes in the nature, speech from people, etc.)

$\mathbf{X} = \{x_1, x_2, ..., x_N\}$
$\mathbf{Y} = \{y_1, y_2, ..., y_N\}$

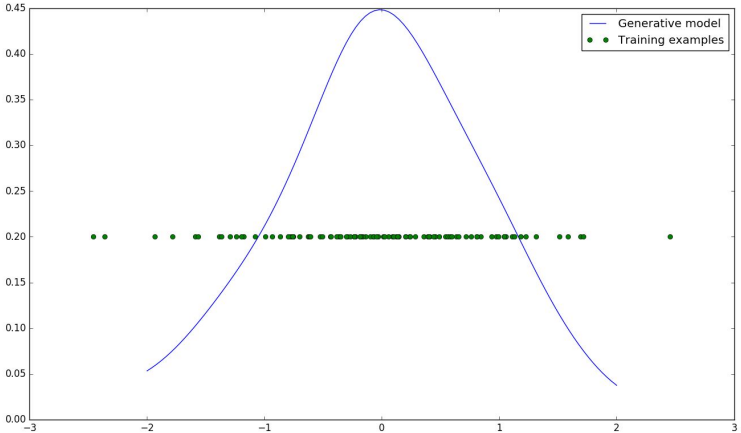
We can also modeled joint probabilities to apply conditioning variables on the generative process: P(X, Y)



Having our dataset **X** with example datapoints (images, waveforms, written text, etc.) each point $\mathbf{x}_n$ lays in some M-dimensional space. Each point $\mathbf{y}_n$ lays in some K-dimensional space. M can be different than K.

What is a generative model?

1) We want our model with parameters ***θ={weights, biases}*** to output samples distributed ***Pmodel***, matching the distribution of our training data ***Pdata*** or P(**X**).
2) We can sample points from ***Pmodel*** plausibly looking ***Pdata*** distributed.

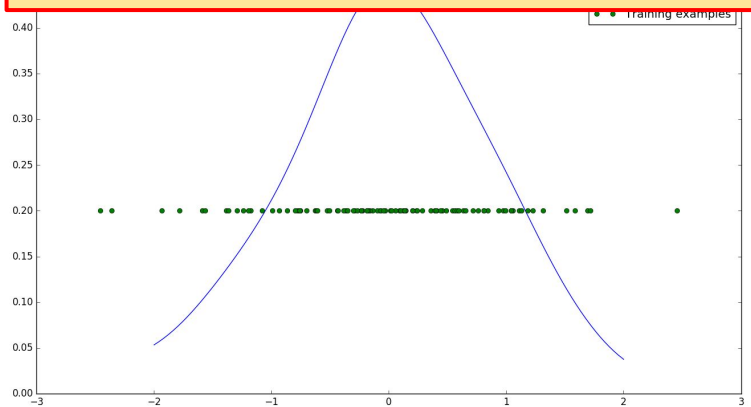


What is a generative model?

1) We want our model with parameters ***θ={weights, biases}*** to output samples distributed ***Pmodel***, matching the distribution of our training data ***Pdata*** or P(**X**).

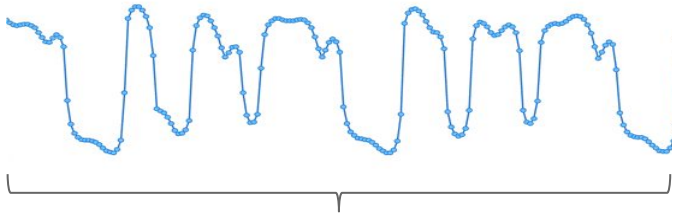
2) We can sampl***Pdata*** distributed.

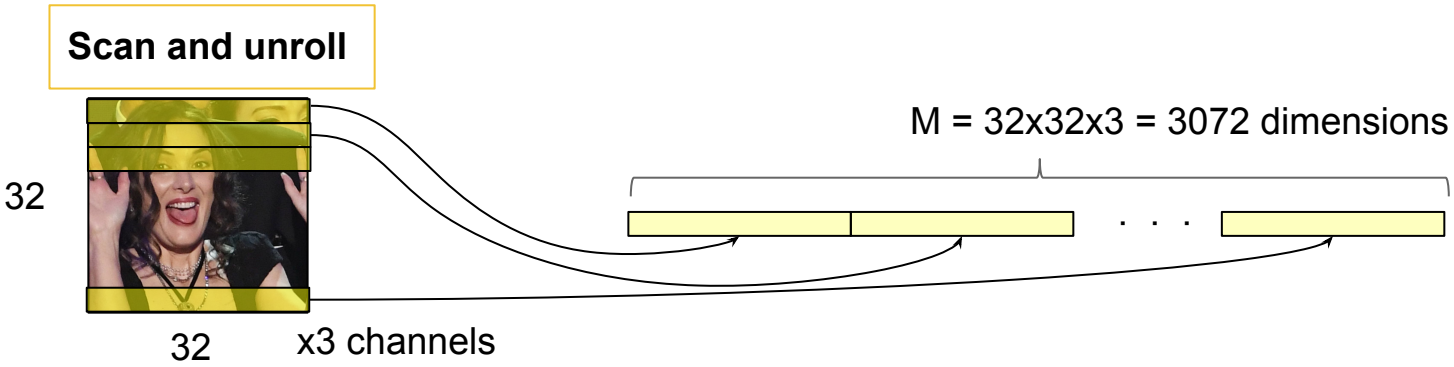
We have not mentioned any network structure or model input data format, just a requirement on what we want in the output of our model.



What is a generative model?

We can generate any type of data, like speech waveform samples or image pixels.





What is a generative model?

Our learned model should be able to make up new samples from the distribution, not just copy and paste existing samples!

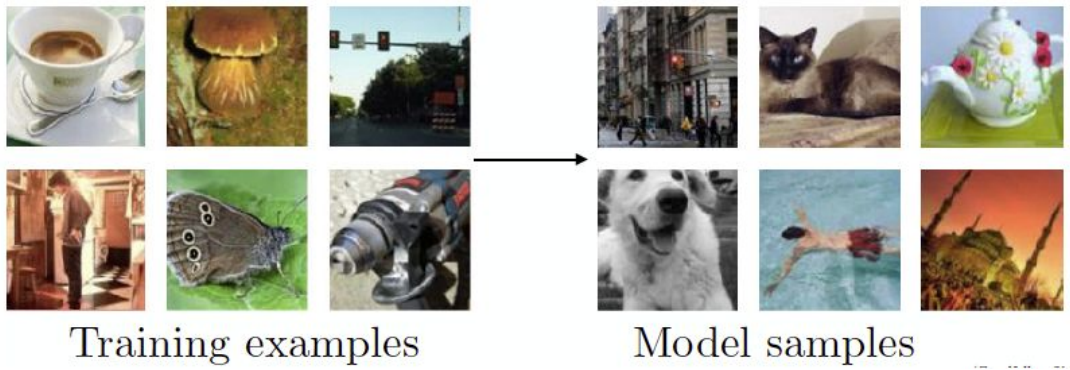


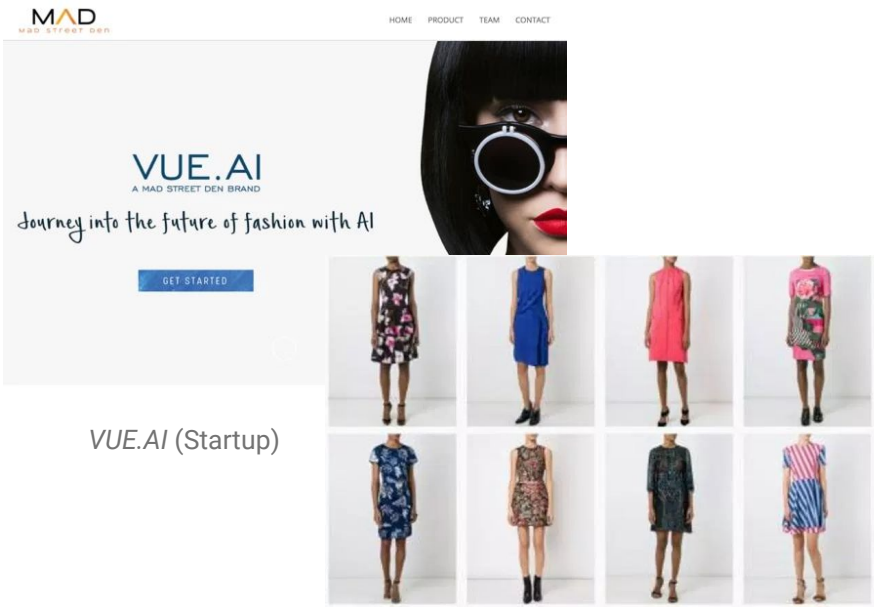
Figure from NIPS 2016 Tutorial: Generative Adversarial Networks (I. Goodfellow)

Why Generative Models?

- Model very complex and high-dimensional distributions.
- Be able to generate realistic synthetic samples
  - possibly perform data augmentation
  - simulate possible futures for learning algorithms
- Unsupervised learning: fill blanks in the data
- Manipulate real samples with the assistance of the generative model
  - Example: edit pictures with guidance (photoshop super pro level)

Motivating Applications

Content Generation



VUE.AI (Startup)

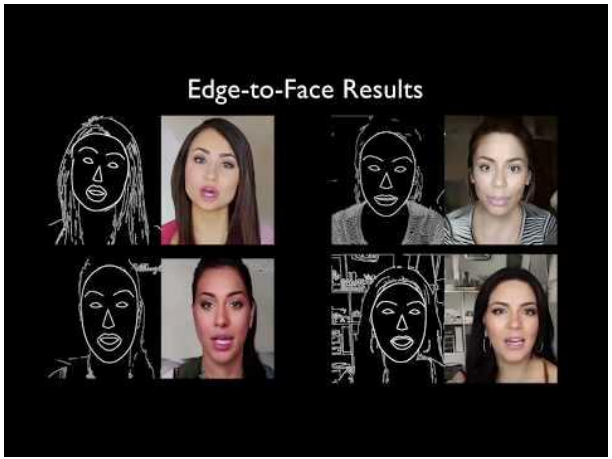
A style-based generator architecture for generative adversarial networks
T. Karras, S. Laine, T. Aila

Image Inpainting

Temporally Coherent GANs for Video Super-Resolution
M. Chu, Y. Xie, L. Leal-Taixe, N. Thuerey

Globally and Locally Consistent Image Completion
S. Iizuka, E. Simo-Serra, H. Ishikawa

Content Manipulation

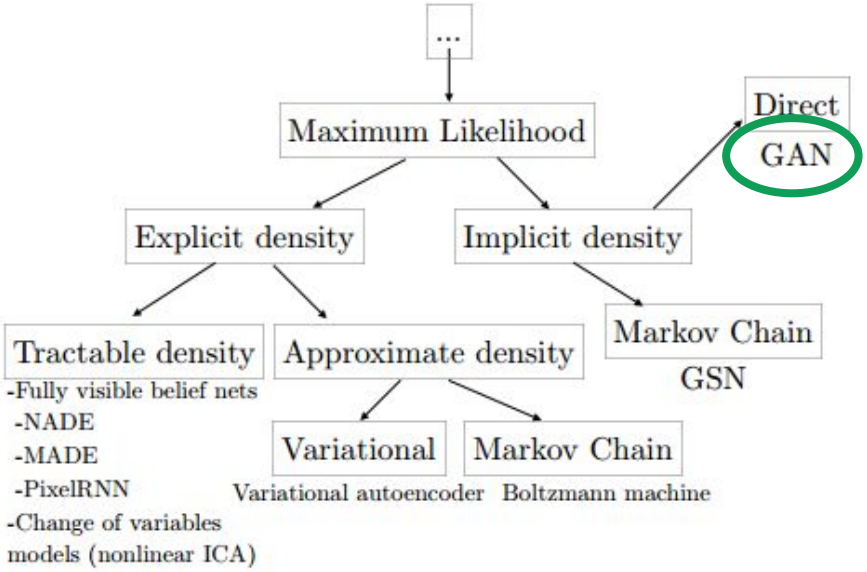


Video-to-Video Synthesis
T. Wang, M. Liu, J. Zhu, G. Liu, A. Tao, J. Kautz, B. Catanzaro

Semantic Image Synthesis with Spatially-Adaptive Normalization
T. Park, M. Liu, T. Wang, J. Zhu

Generative Models Taxonomy

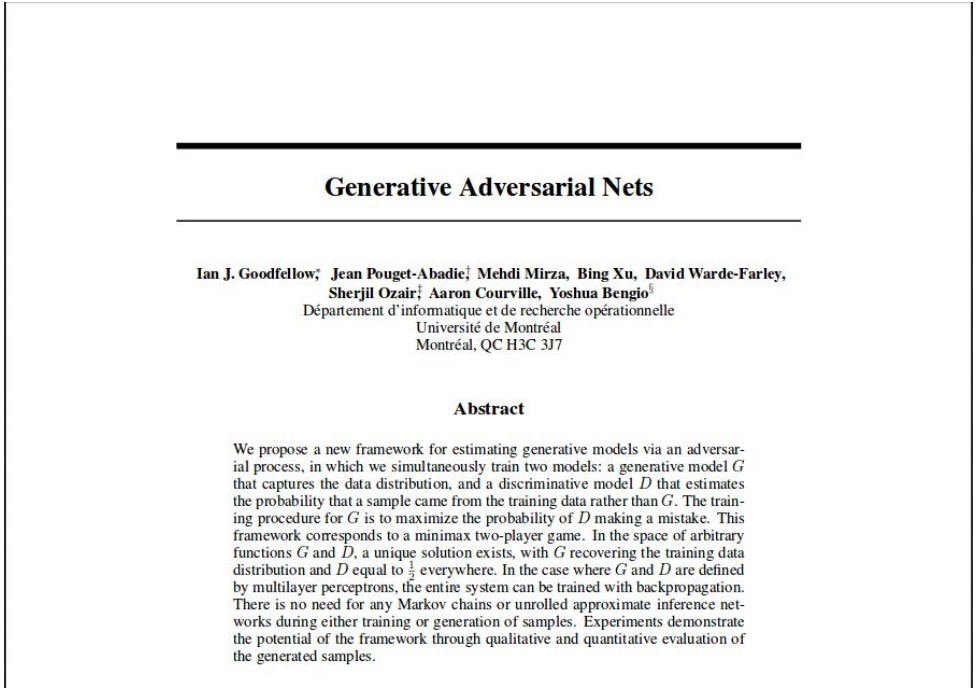
Taxonomy


...
Maximum Likelihood
Direct
GAN
Explicit density
Implicit density
Markov Chain
GSN
Tractable density
Approximate density
-Fully visible belief nets
-NADE
-MADE
-PixelRNN
-Change of variables
models (nonlinear ICA)
Variational
Markov Chain
Variational autoencoder
Boltzmann machine

Generative Adversarial Networks

First GANs appearance, back then… (only 5 years ago)

(Goodfellow et al. 2014)



Generative Adversarial Nets

Ian J. Goodfellow[*], **Jean Pouget-Abadie**[†], **Mehdi Mirza**, **Bing Xu**, **David Warde-Farley**, **Sherjil Ozair**[†], **Aaron Courville**, **Yoshua Bengio**[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

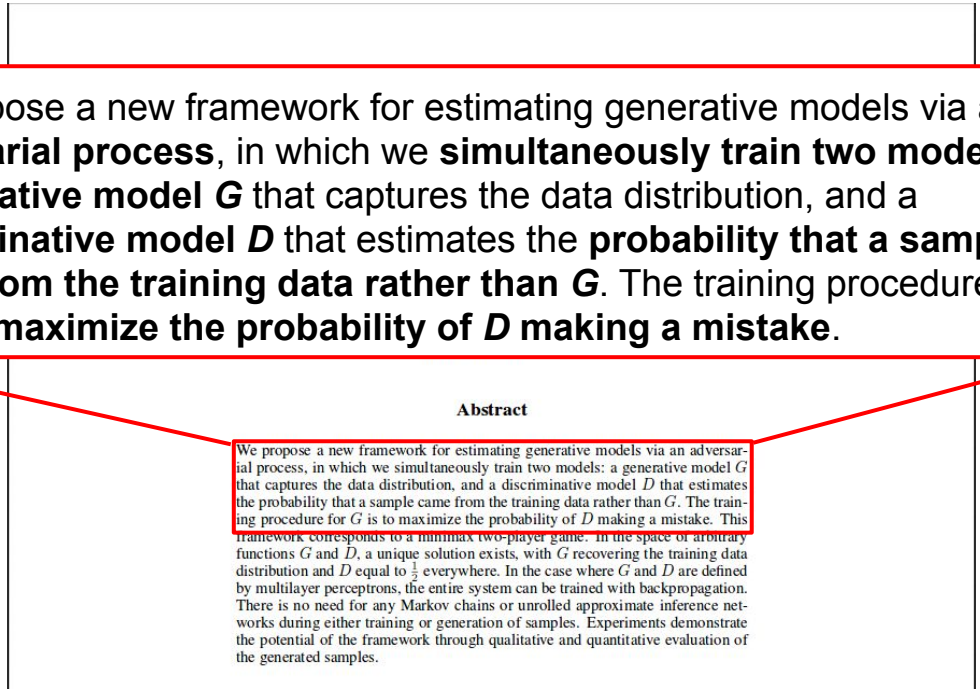
Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

First GANs appearance, back then… (only 5 years ago)

(Goodfellow et al. 2014)

We propose a new framework for estimating generative models via an **adversarial process**, in which we **simultaneously train two models**: a **generative model *G*** that captures the data distribution, and a **discriminative model *D*** that estimates the **probability that a sample came from the training data rather than *G***. The training procedure for ***G* is to maximize the probability of *D* making a mistake**.

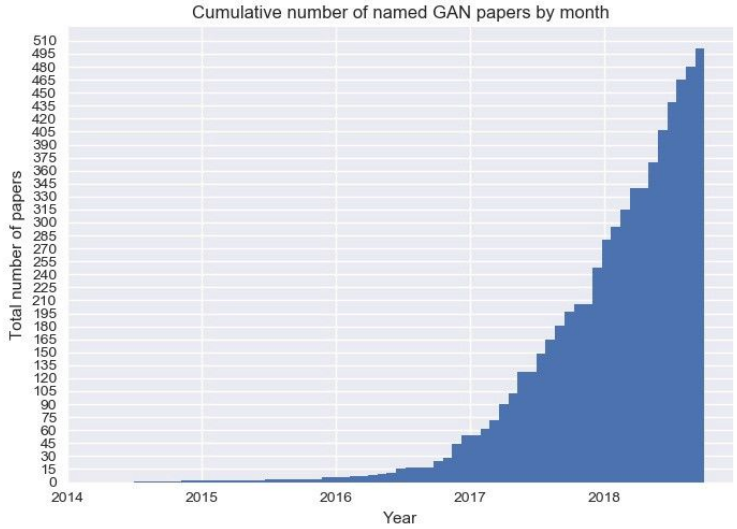


Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

GANs are coming in DGMs II ...

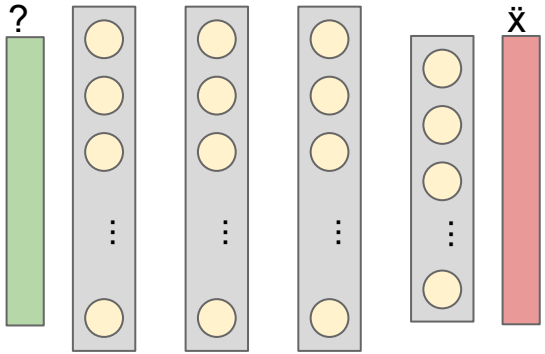
The GAN epidemic



GAN schematic

Generator network *G* → generate samples ẍ that follow a probability distribution Pg, and make it close to Pdata (our dataset samples distribution).

Q: What do we input to our model in order to make up new samples through a neural network?



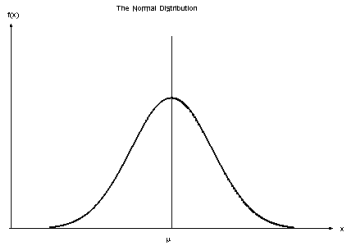
Generated Samples

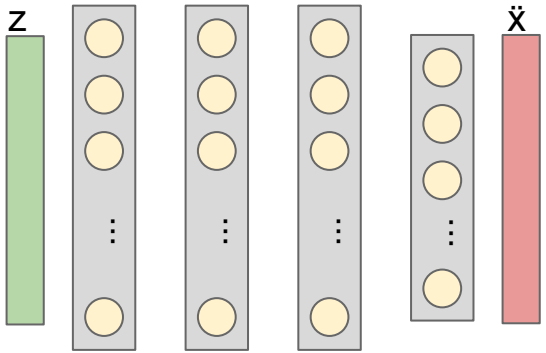
GAN schematic

Generator network *G* → generate samples ẍ that follow a probability distribution Pg, and make it close to Pdata (our dataset samples distribution).

Q: What do we input to our model in order to make up new samples through a neural network?

A: We can sample from a simplistic prior, for example, a Gaussian distribution N(0, I).





Generated Samples

GAN schematic

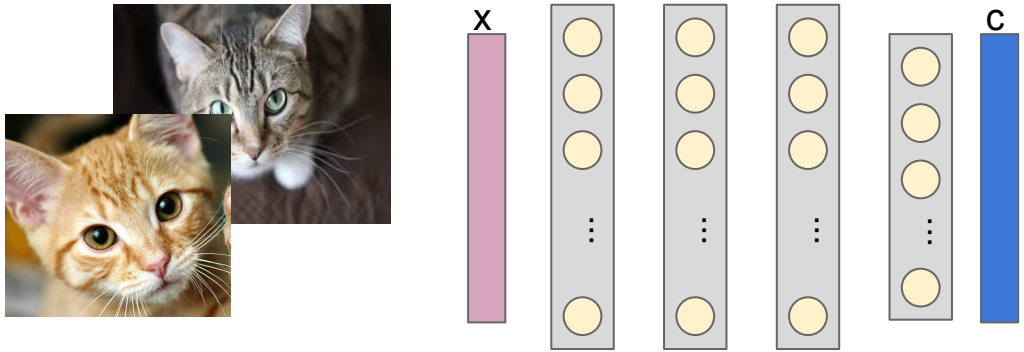
Q: How can we evaluate the quality of the generated samples?

GAN schematic

Q: How can we evaluate the quality of the generated samples?

A: With another network trained to do so.

Discriminator network *D* → probability that a sample came from the training data rather than *G*.

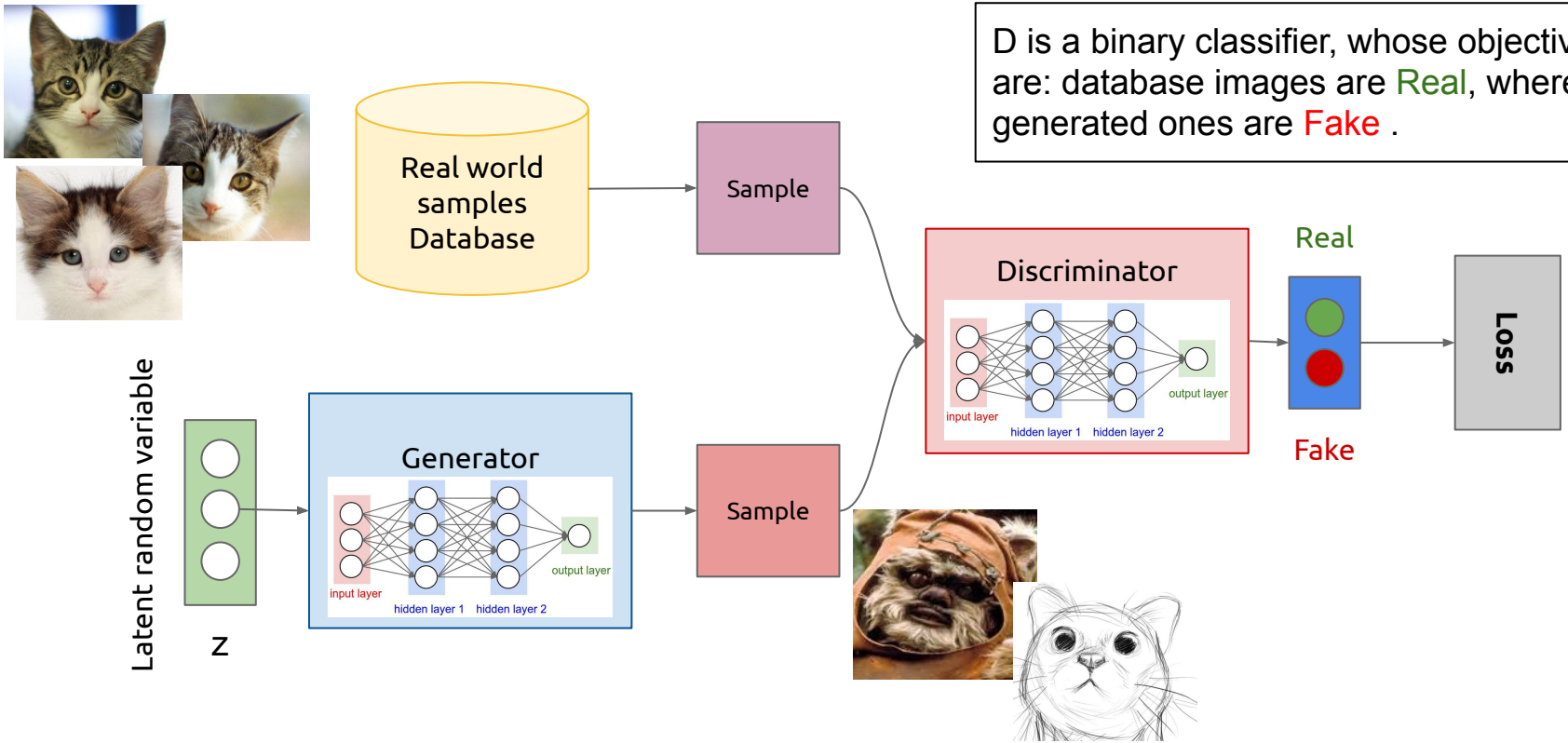


Generative + Adversarial

We have two modules: **Generator** (G) and **Discriminator** (D).

- They “fight” against each other during training→ **Adversarial Learning**
- G mission: Fool D to missclassify.
- *D* mission: Discriminate between *G* samples and real samples.

Generative + Adversarial



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator training

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator training

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

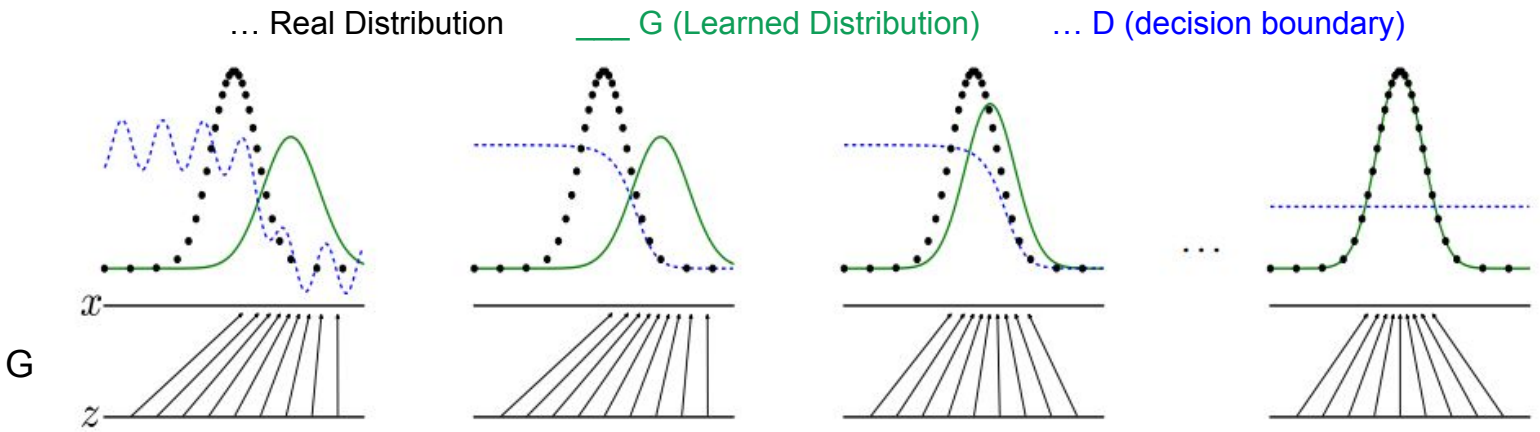
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Training GANs dynamics

Iterate these two steps until convergence (which may not happen)

- Updating the discriminator should make it better at discriminating between real images and generated ones (**discriminator improves**)
- Updating the generator makes it better at fooling the current discriminator (**generator improves**)

Eventually (we hope) that the generator gets so good that it is impossible for the discriminator to tell the difference between real and generated images. Discriminator accuracy = 0.5

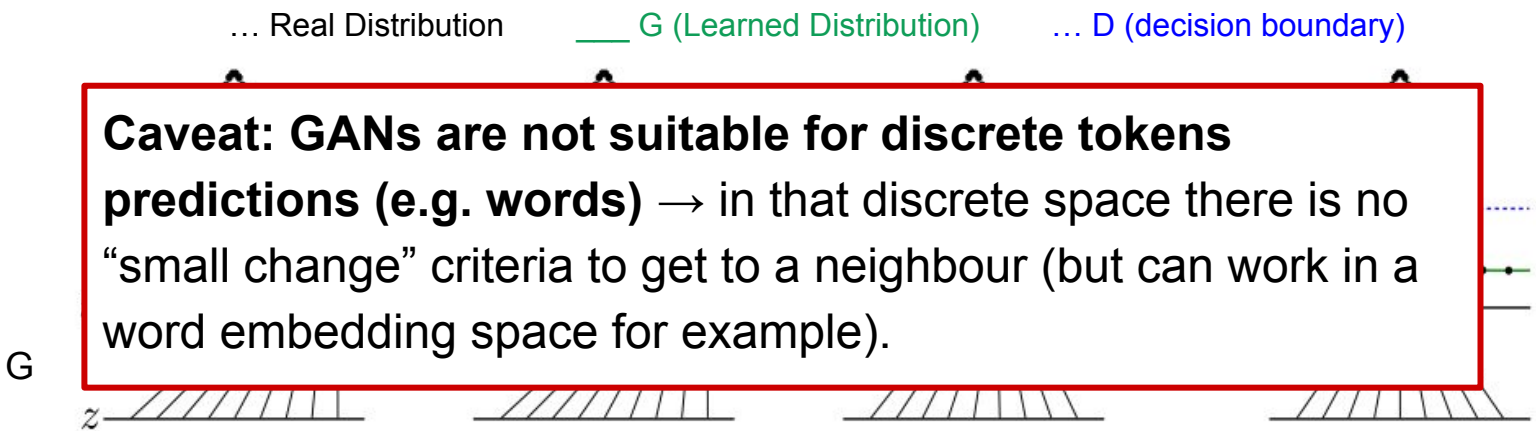


Training GANs dynamics

Iterate these two steps until convergence (which may not happen)

- Updating the discriminator should make it better at discriminating between real images and generated ones (**discriminator improves**)
- Updating the generator makes it better at fooling the current discriminator (**generator improves**)

Eventually (we hope) that the generator gets so good that it is impossible for the discriminator to tell the difference between real and generated images. Discriminator accuracy = 0.5



Caveat: GANs are not suitable for discrete tokens predictions (e.g. words) → in that discrete space there is no "small change" criteria to get to a neighbour (but can work in a word embedding space for example).

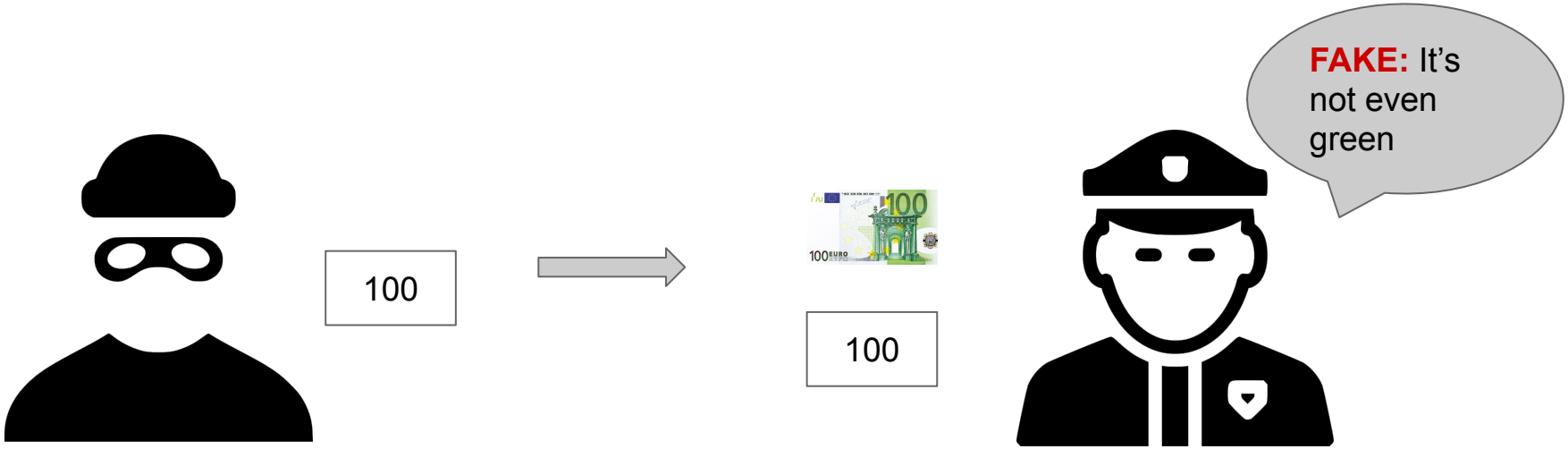
Adversarial Training Analogy: is it fake money?

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.

Key Idea: D is trained to detect fraud, (its parameters learn discriminative features of “what is real/fake”). As backprop goes through **D** to **G** there happens to be information leaking about the requirements for bank notes to look real. This makes **G** perform small corrections to get closer and closer to what real samples would be.

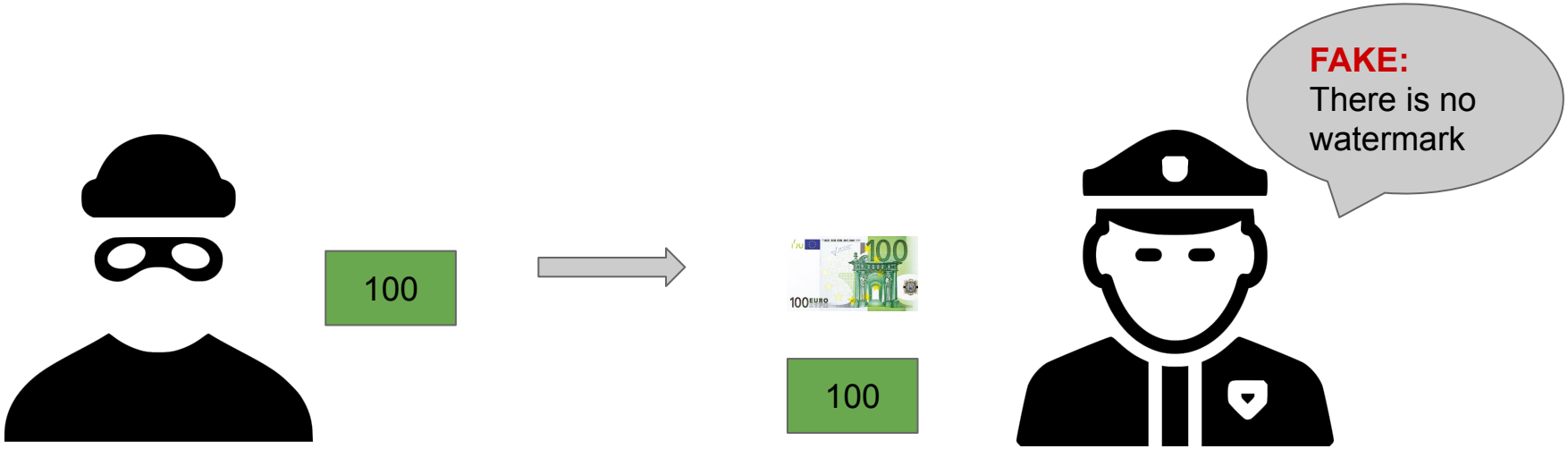
Adversarial Training Analogy: is it fake money?

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.



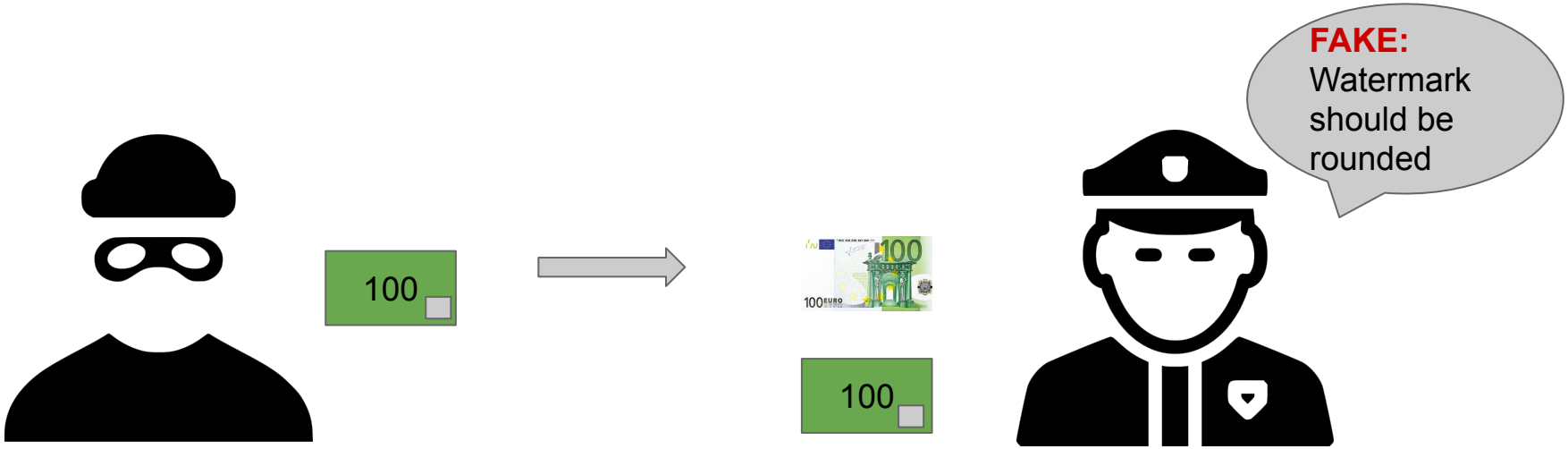
Adversarial Training Analogy: is it fake money?

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.



Adversarial Training Analogy: is it fake money?

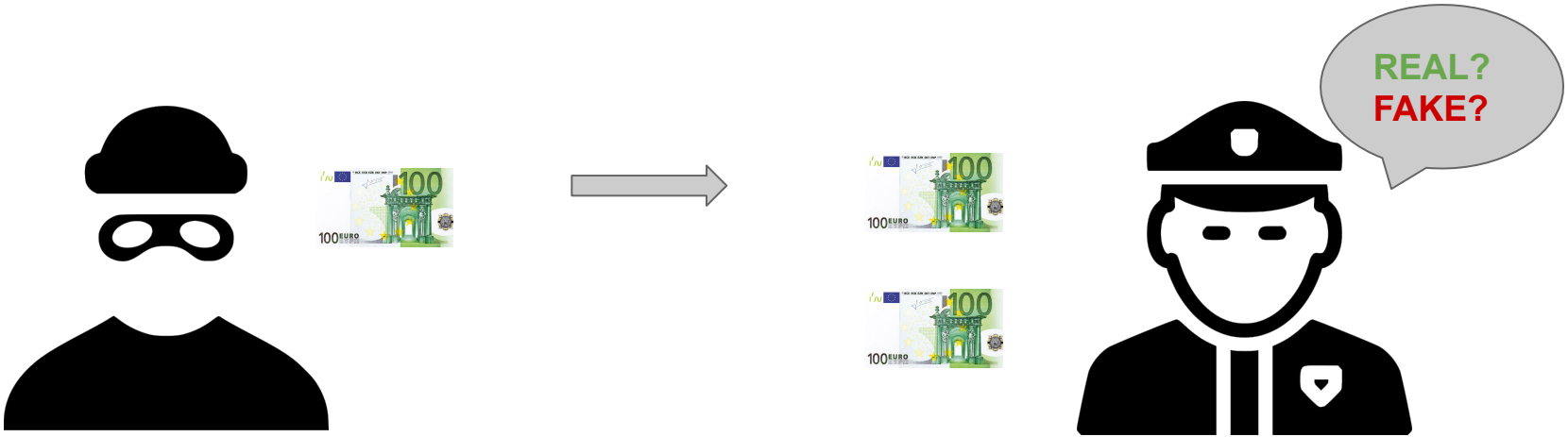
Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.



Adversarial Training Analogy: is it fake money?

Imagine we have a counterfeiter (**G**) trying to make fake money, and the police (**D**) has to detect whether money is real or fake.

After enough iterations, and if the counterfeiter is good enough (in terms of **G** network it means “has enough parameters”), the police should be confused.



Formulations

We have networks **G** and **D**, and **training set with pdf *Pdata***. Notation:

- ***θ(G)***, ***θ(D)*** (Parameters of model **G** and **D** respectively)
- ***x*** ~ Pdata (M-dim sample from training data pdf)
- ***z*** ~ *N(0, I)* (sample from prior pdf, e.g. N-dim normal)
- **G(z) = ẍ** ~ Pmodel (M-dim sample from G network)

D network receives **x** or **ẍ** inputs → decides whether input is real or fake. It is **optimized to learn: x is real (1), ẍ is fake (0) (binary classifier).**

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

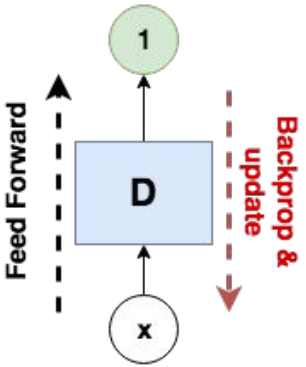
G network maps sample **z** to G(**z**) = **ẍ** → it is **optimized to maximize D mistakes.**

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

NIPS 2016 Tutorial: Generative Adversarial Networks. Ian Goodfellow

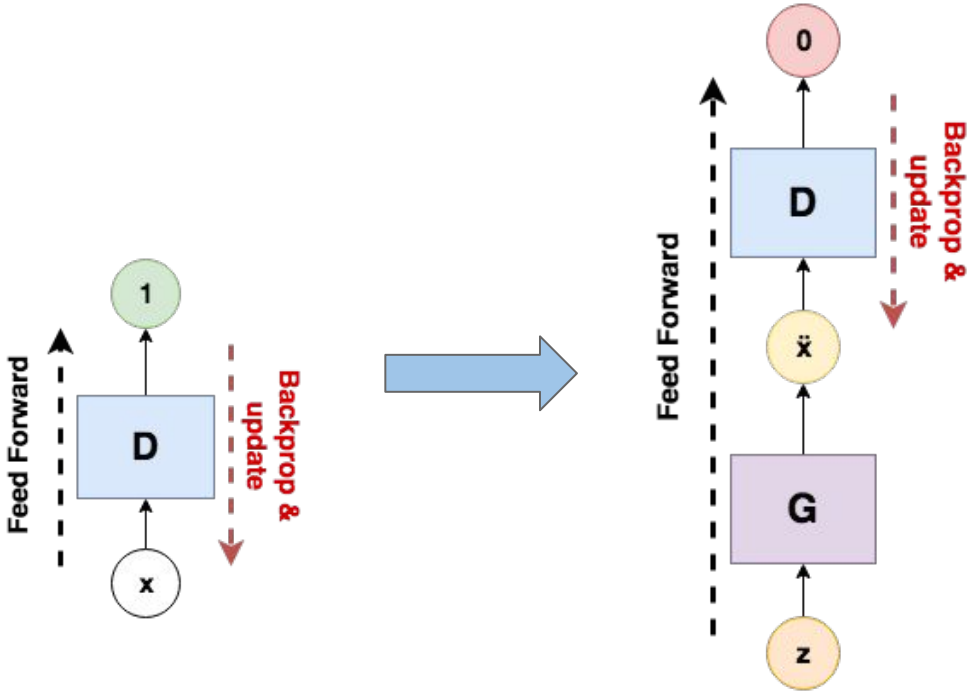
Adversarial Training (batch update) (1)

- Pick a sample ***x*** from training set
- Show ***x*** to **D** and update weights to output 1 (real)



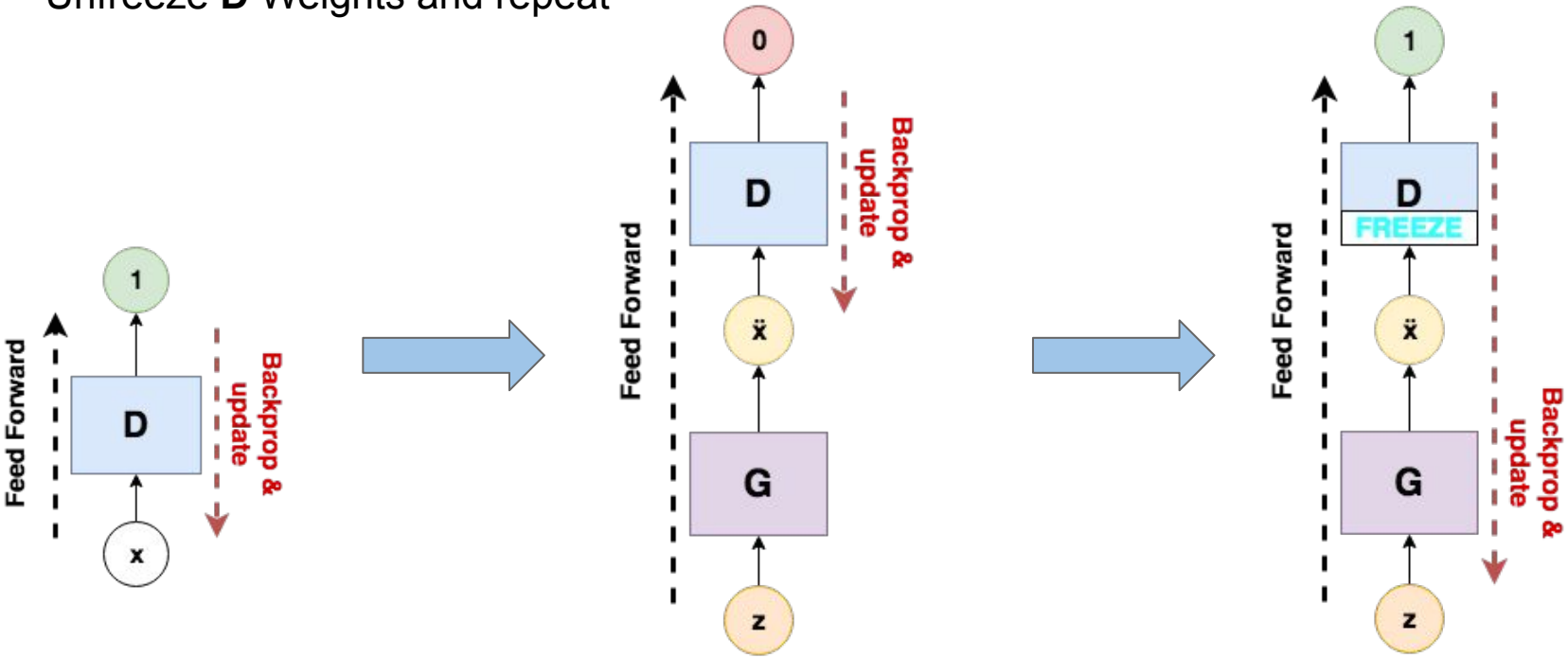
Adversarial Training (batch update) (2)

- **G** maps sample ***z*** to ***ẍ***
- show ***ẍ*** and update weights to output 0 (fake)



Adversarial Training (batch update) (3)

- Freeze **D** weights
- Update **G** weights to make **D** output 1 (just **G** weights!)
- Unfreeze **D** Weights and repeat

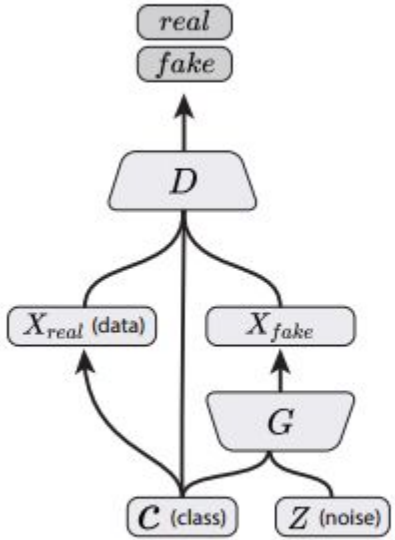


Conditional GANs

For details on ways to condition GANs: Ways of Conditioning Generative Adversarial Networks (Wack et al.)

GANs can be conditioned on other info extra to **z**: text, labels, speech, etc..

z might capture random characteristics of the data (variabilities of plausible futures), whilst **c** would condition the deterministic parts !



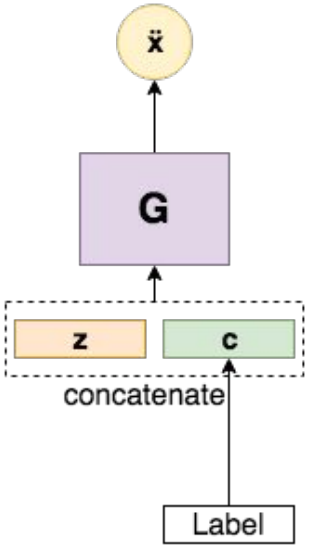
Conditional GAN
(Mirza & Osindero, 2014)

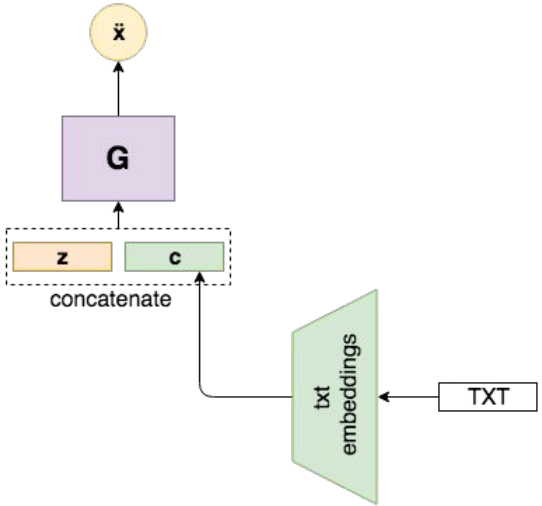
Conditional GANs

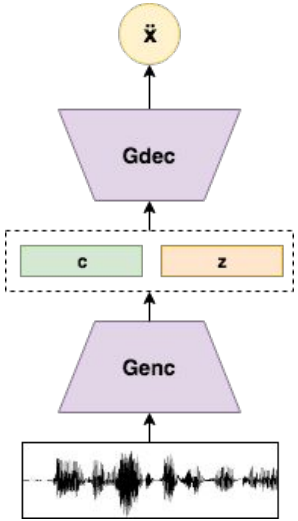
For details on ways to condition GANs: [Ways of Conditioning Generative Adversarial Networks (Wack et al.)]

GANs can be conditioned on other info extra to **z**: text, labels, speech, etc..

z might capture random characteristics of the data (variabilities of plausible futures), whilst **c** would condition the deterministic parts !







Caveats

Where is the downside...?

GANs are tricky and hard to train! We do not want to minimize a cost function. Instead **we want both networks to reach Nash equilibria (saddle point)**.

- Formulated as a “game” between two networks
- Unstable dynamics: hard to keep generator and discriminator in balance
- Optimization can **oscillate** between solutions
- Generator can collapse (generate low variability at its output)

Caveats

Where is the downside...?

GANs are tricky and hard to train! We do not want to minimize a cost function. Instead **we want both networks to reach Nash equilibria (saddle point)**.

Because of extensive experience within the GAN community (with some does-not-work-frustration from time to time), you can find some tricks and tips on how to train a vanilla GAN here:
https://github.com/soumith/ganhacks

Evolving GANs

Different loss functions in the Discriminator can fit your purpose:

- Binary cross-entropy (BCE): Vanilla GAN
- Least-Squares: LSGAN
- Wasserstein GAN + Gradient Penalty: WGAN-GP
- BEGAN: Energy based (D as an autoencoder) with boundary equilibrium.
- Maximum margin: Improved binary classification

Least Squares GAN

Least Squares Generative Adversarial Networks, Mao et al. 2016

Main idea: shift to loss function that provides smooth & non-saturating gradients in D

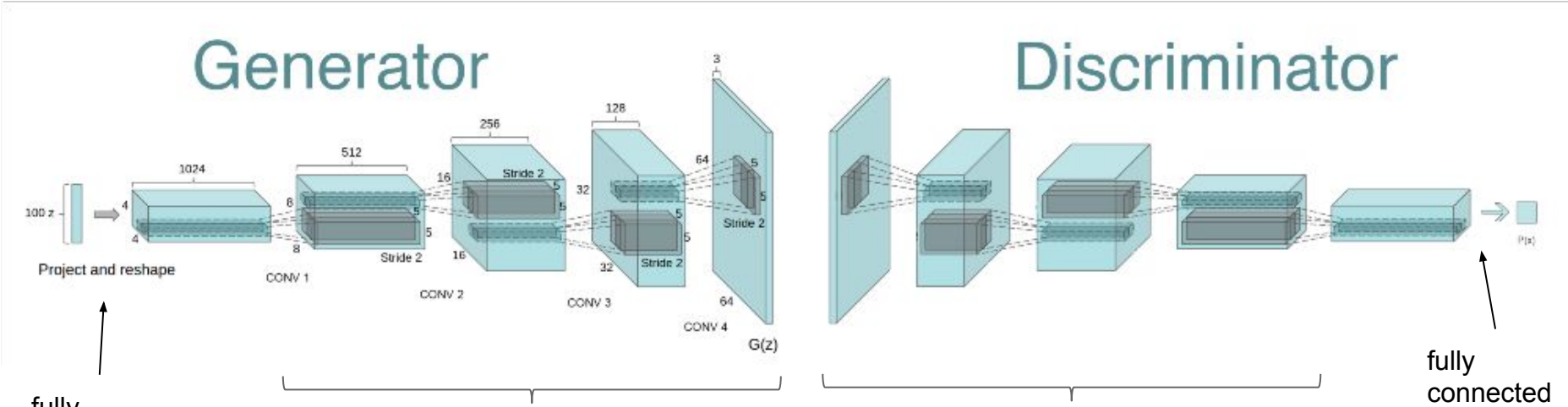
- **Because of sigmoid saturation** in binary classification loss, G gets no info when D gets to label true examples → **vanishing gradients make G no learn**
- Least squares loss improves learning with notion of distance of *Pmodel* to *Pdata*:

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right],$$

Generating images/frames

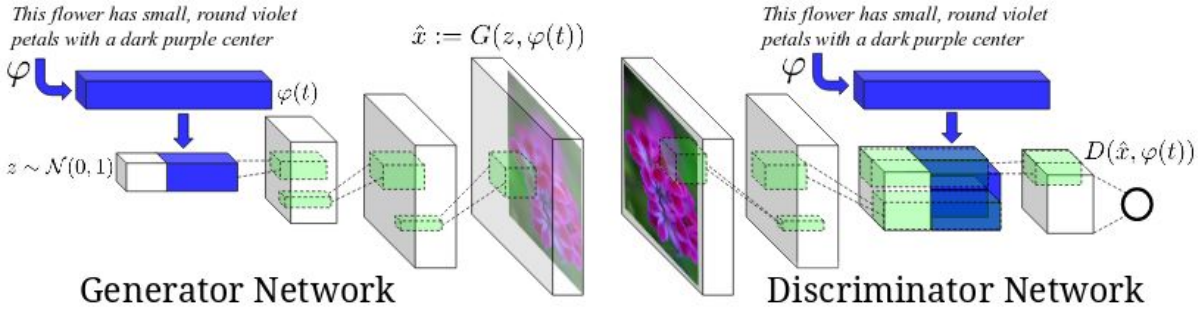
Deep Conv. GAN (DCGAN) first to effectively generat 64x64 RGB images in a single shot. It is also the base of all image generation GAN architectures.



(Radford et al. 2015)

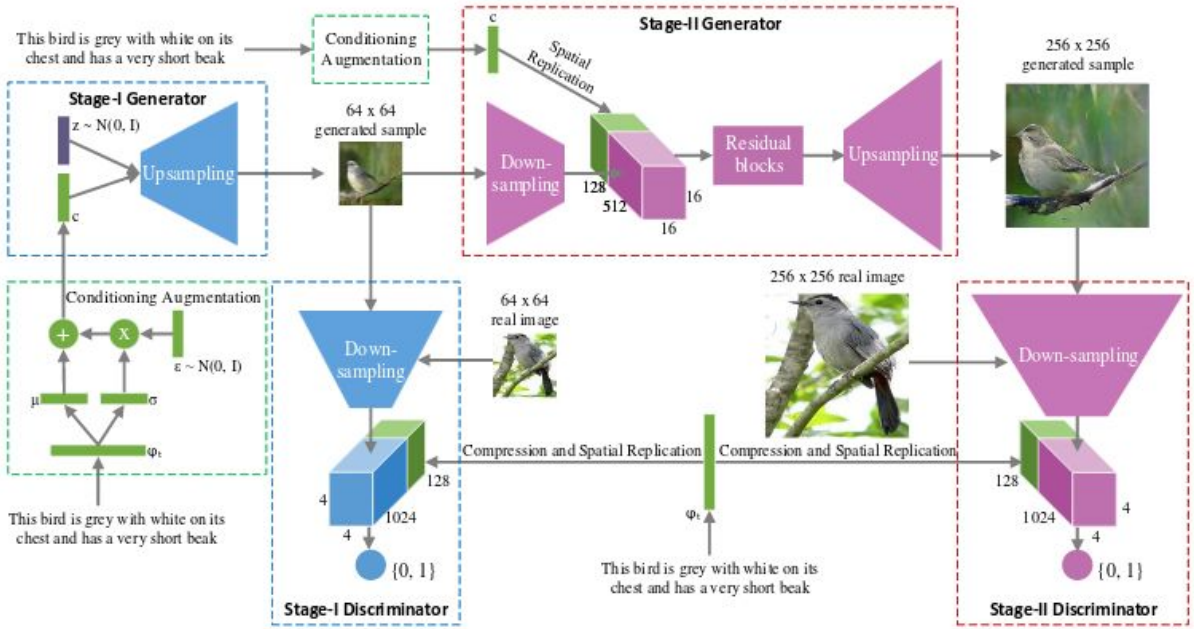
Generating images/frames conditioned on captions

Generative Adversarial Text to Image Synthesis (Reed et al. 2016b)

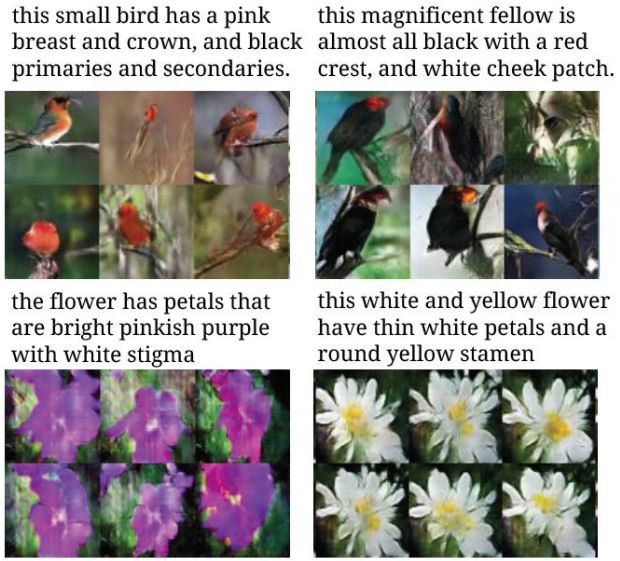


Generating images/frames conditioned on captions

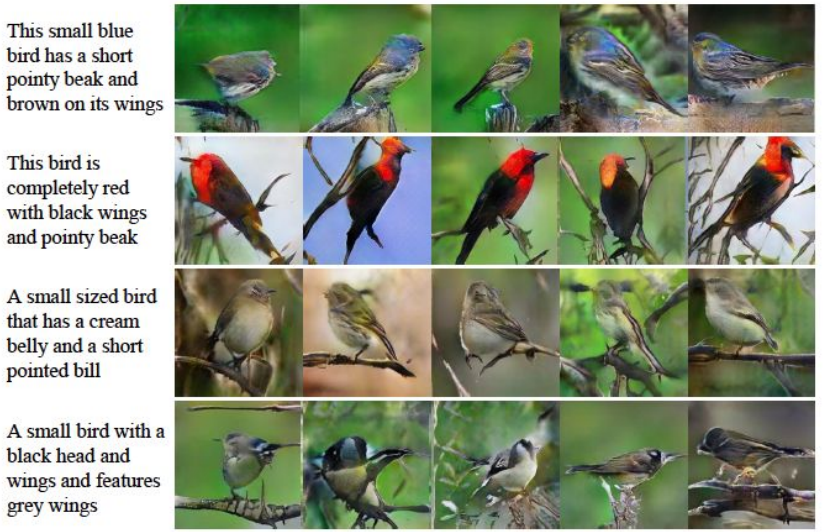
StackGAN (Zhang et al. 2016). Increased resolution to 256x256, conceptual two-stage: ‘Draft’ and ‘Fine-grained detalis’



Generating images/frames conditioned on captions



(Reed et al. 2016b)

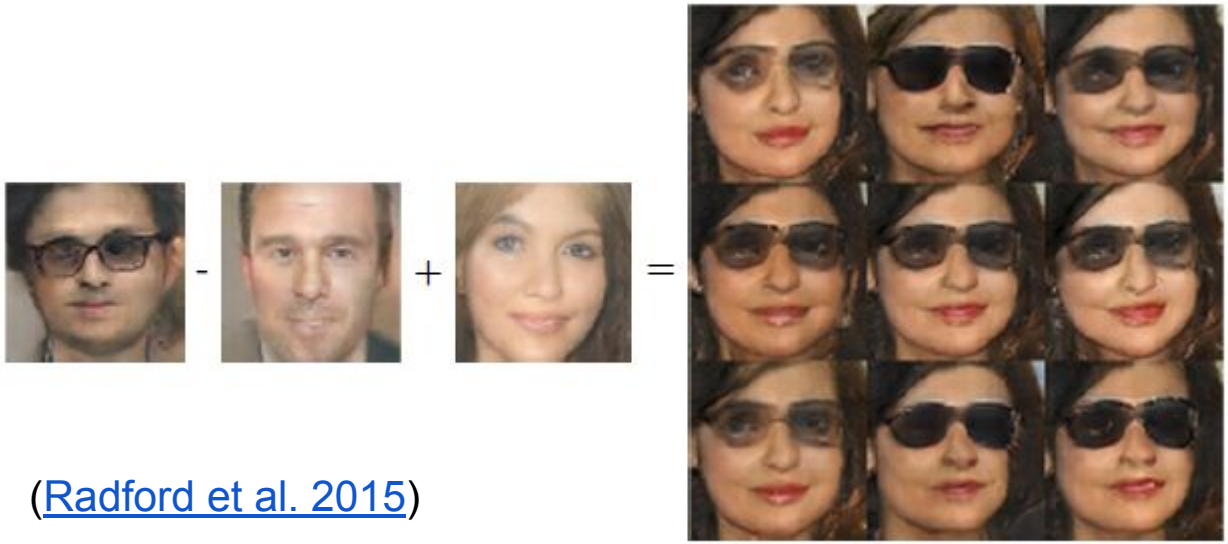


(Zhang et al. 2016)

Unsupervised feature extraction/learning representations

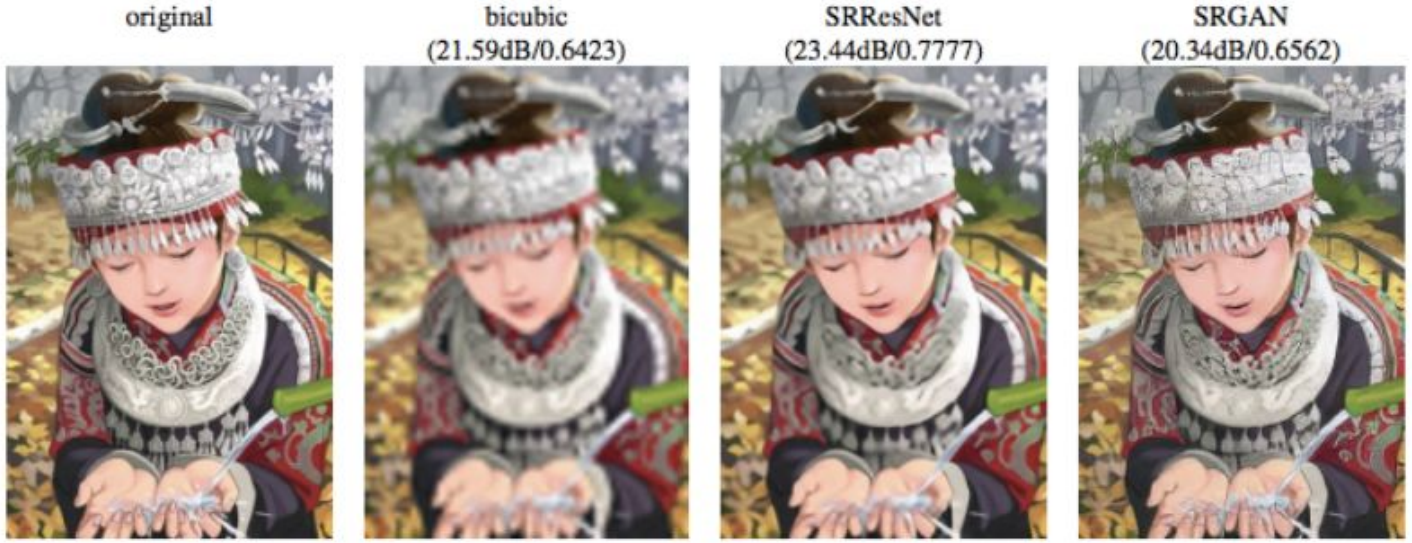
Similarly to word2vec, GANs learn a distributed representation that disentangles concepts such that we can perform semantic operations on the data manifold:

v(Man with glasses) - v(man) + v(woman) = v(woman with glasses)



(Radford et al. 2015)

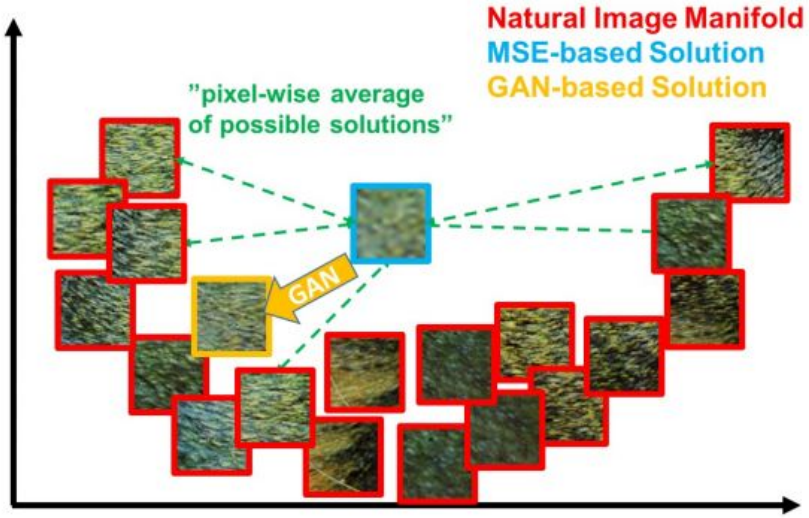
Image super-resolution



(Ledig et al. 2016)

Bicubic: not using data statistics. SRResNet: trained with MSE. SRGAN is able to understand that there are multiple correct answers, rather than averaging.

Image super-resolution

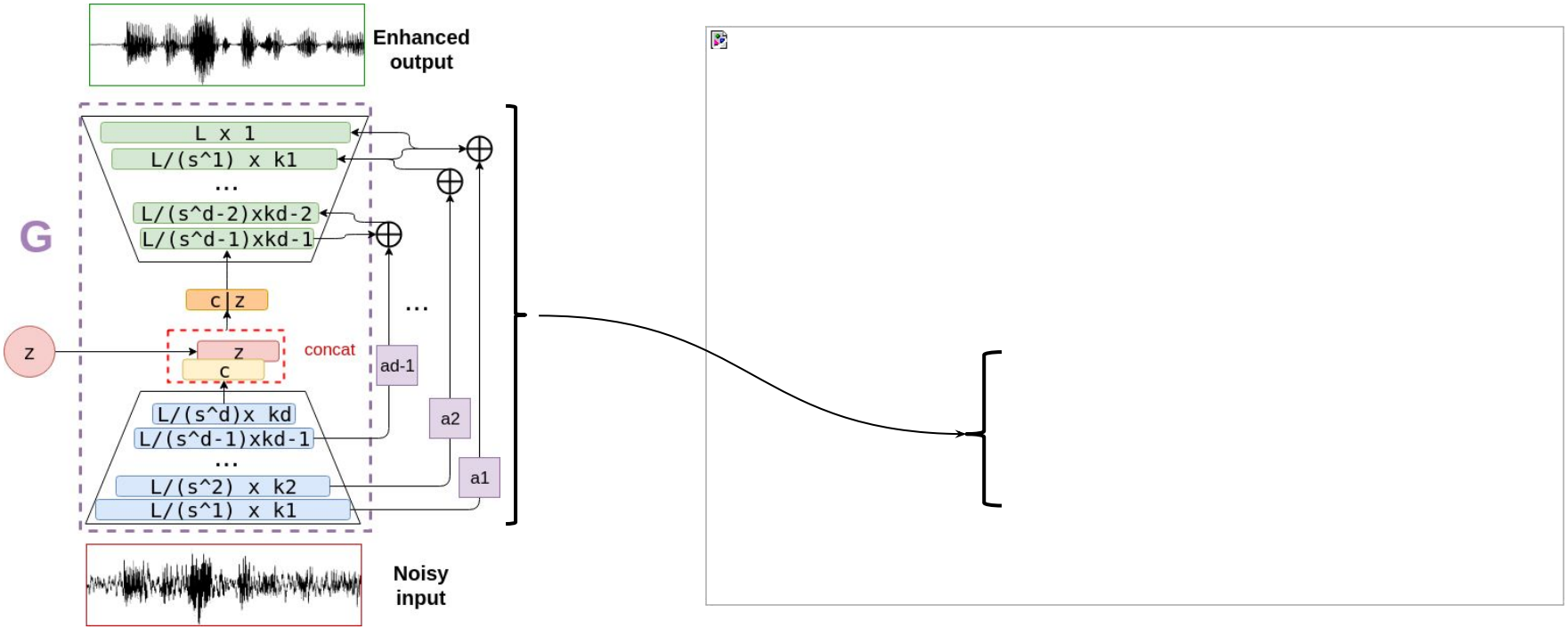


(Ledig et al. 2016)

Averaging is a serious problem we face when

Speech Enhancement

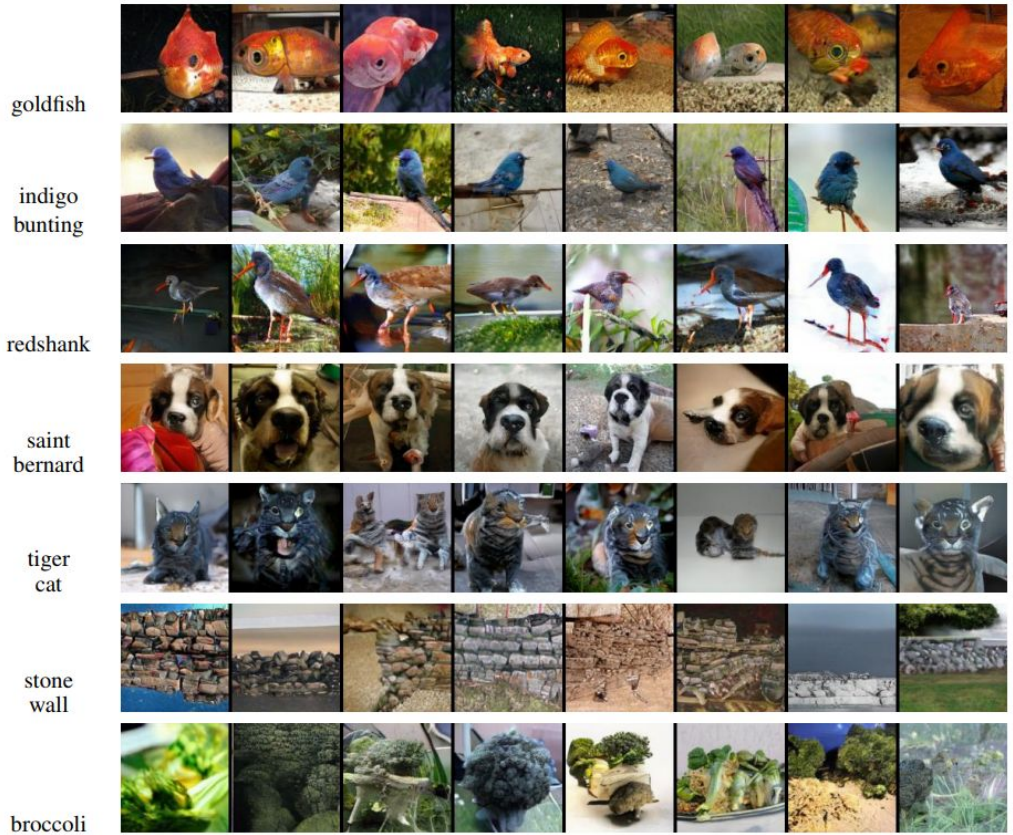
Speech Enhancement GAN (Pascual et al. 2017)



Self-Attention GAN Imagenet conditional generation (2018)

New May 2018 state of the art results with 128x128 imagenet generated images. Global structure is now preserved and GANs traing more stable.

(Zhang et al. 2018)



Big GAN (2018)

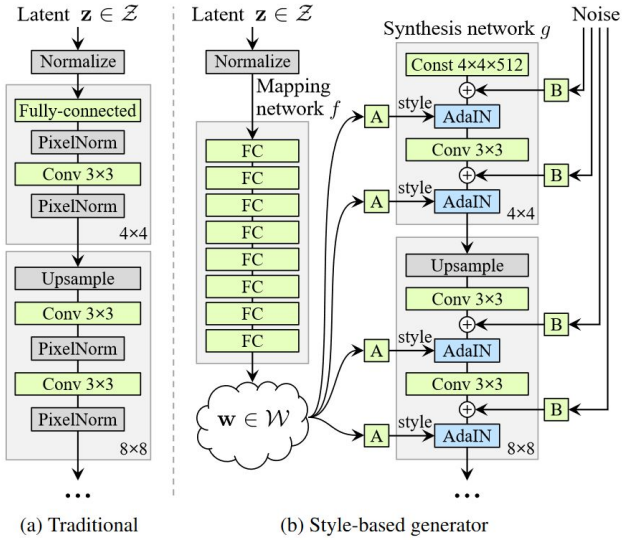
Current (Oct 2018) state of the art results with 512x512 imagenet generated images!

(Brock et al. 2018)

GANimation (2018)

(Pumarola et al. 2018)

StyleGan (2019)



(Karras et al. 2019)

Thanks! Questions?