Lecture 13: Generative Adversarial Models

Madhavan Mukund and Pranabendu Misra

Advanced Machine Learning 2021

Recall: AutoEncoders and VAEs

Learning a sensible and compressed representation of a given dataset is an important task:

- Given an image, learn about the objects in the image
    - Then we can add / remove / modify the objects in the image, thereby generating a new image
    - We can generate similar images
    - We can create sensible interpolation between two images

Recall: AutoEncoders and VAEs

Learning a sensible and compressed representation of a given dataset is an important task:

- Given an image, learn about the objects in the image
  - Then we can add / remove / modify the objects in the image, thereby generating a new image
  - We can generate similar images
  - We can create sensible interpolation between two images

- Given an music file, learn about the instruments in the music
  - Remove noise and distortions
  - Add or remove other instruments
  - and so on ...

This is an unsupervised learning task

- Interpolation between the two images would be:

- Example: shifting the image

- Example: Night to Day

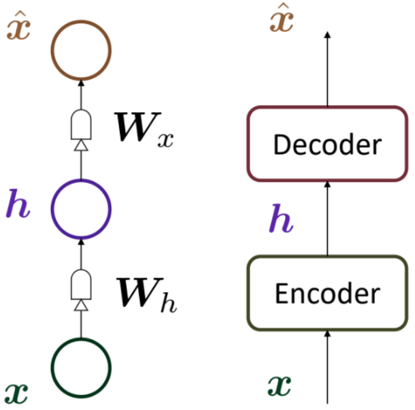
- Super Resolution

- Caption to Image:

Input Caption: A bright blue bird with white belly

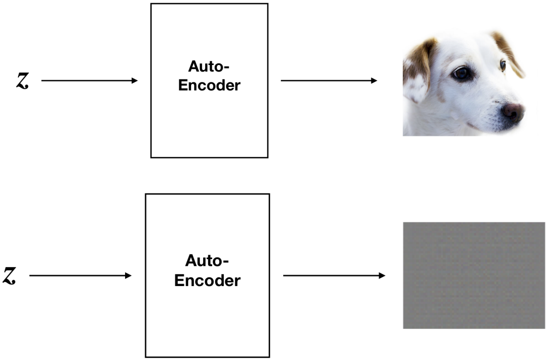
Autoencoders

- The idea is to learn how to encode the input into a compressed form, and then re-generate the input from the encoding
- An encoder network learns how to map the input x to a code h in the latent space of far smaller dimension.
- And at the same time, a decoder network learns to map the code h back to an approximate from of the input $\hat{x}$
- This is an *Unsupervised Learning* task.



Autoencoders: Data generation from random point in Latent Space?

Expectation:

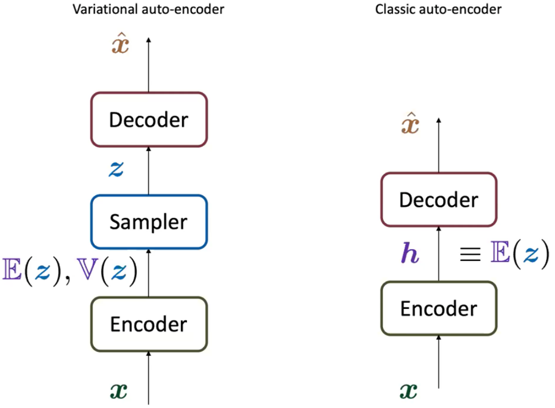


Turns out that auto-encoders learn something like jpeg compression, most random latent-space points are meaning-less.

Variational Autoencoders (VAE)

The generation aspect is at the center of this model.

- The encoder maps x to a code h which has two parts $E(z)$ and $V(z)$.
  E and V stand for expectation and variance
- The next step is to sample a point z fror a *Gaussian distribution* with expectation $E(z)$ and variance $V(z)$.
  $$z = E(z) + \epsilon \odot \sqrt{V(z)}$$
  where $\epsilon \sim N(0, I_d)$ and d is the dimensio of the latent-space.
- Finally, the decoder maps this randomly sampled point z to $\hat{x}$



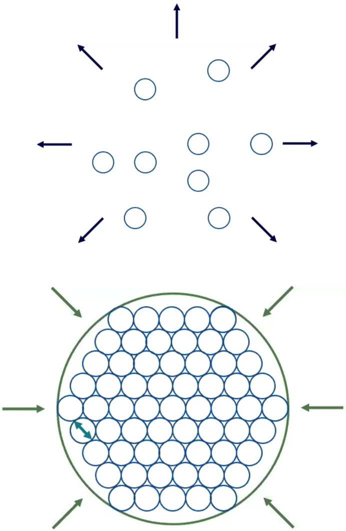
Variational Autoencoders (VAE)

Loss functions for VAE:

- Apart from the usual loss, there is an extra regularization term in the loss

$$\beta \cdot \ell_{KL}(z, N(0, I_d))$$

- The reconstruction error $\ell(x, \hat{x}) = \frac{1}{2}\|x - \hat{x}\|^2$ pushes the "probability balls" apart, because it penalizes overlapping.
  We also have no control over the "size" of the bubbles.
- Introducing regularization term in the loss counters this.



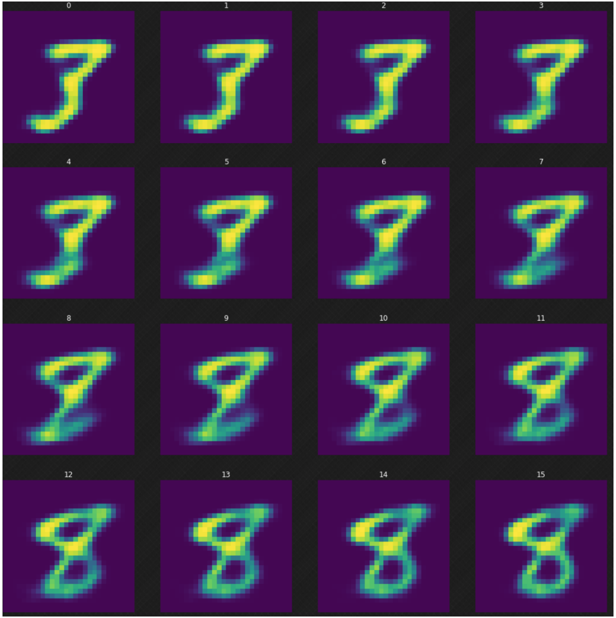
Examples: VAE trained on Celebrity Faces

Interpolating between hair-colors:

Examples: VAE trained on Celebrity Faces

Interpolating between hair-colors:

Interpolating between no glasses and glasses


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Generative Adversarial Network

- Another class of *Generative* models.

Generative Adversarial Network

- Another class of *Generative* models.
- **The generation step is at the heart of this model**.

- Another class of *Generative* models.
- **The generation step is at the heart of this model**.

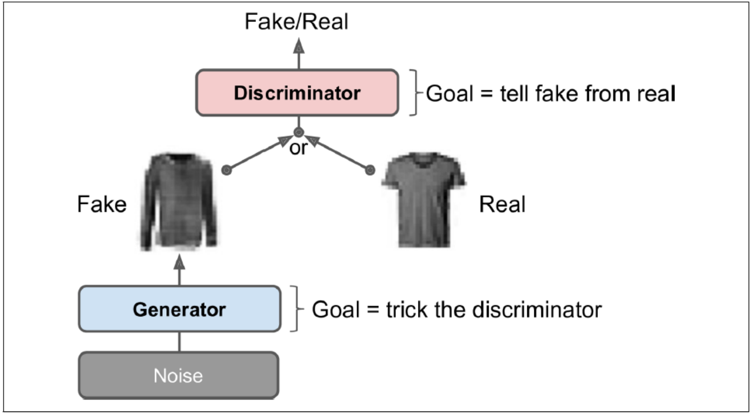


Figure 17-15. A generative adversarial network

Generative Adversarial Network

- **Generator**: A Neural Network which given some *random noise* it generates a *fake datapoint*.
- **Discriminator**: A Neural Network that given an image determines if it is real or fake.

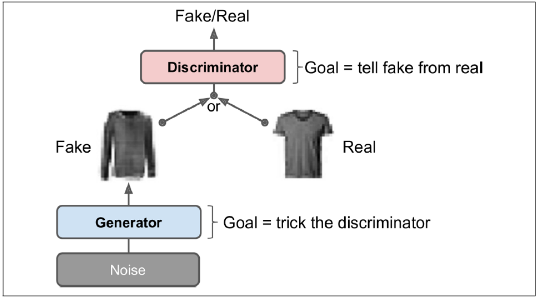
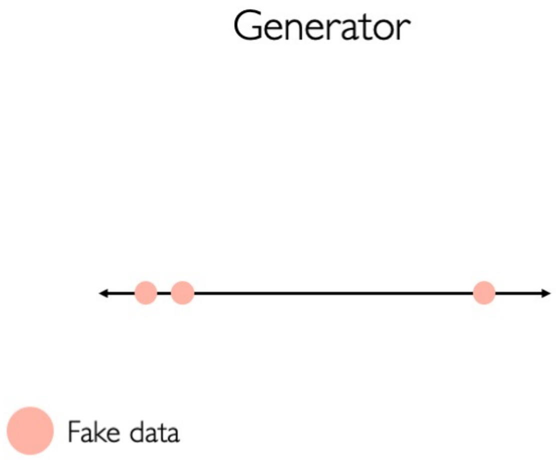


Figure 17-15. A generative adversarial network

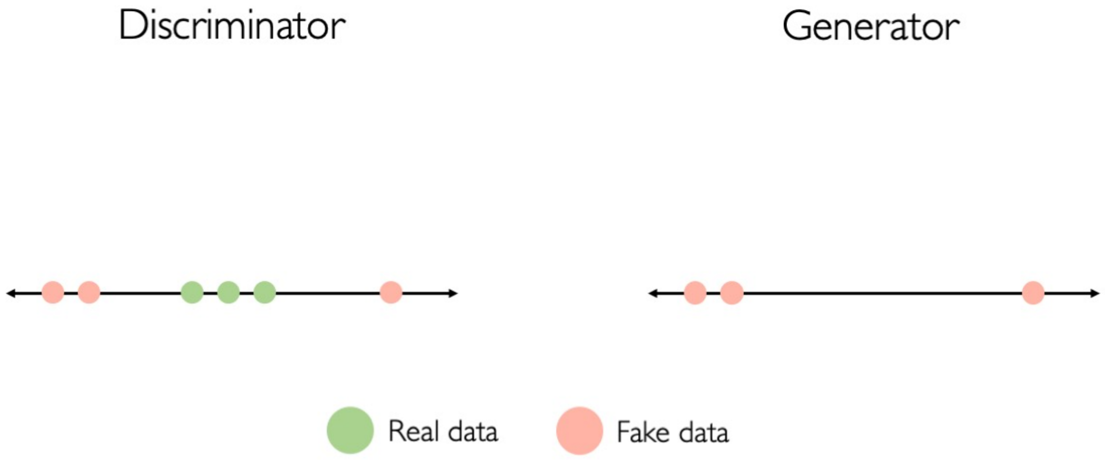
This is a game-theoretic modeling of the problem.

- Generator and Discriminator are both trying to outwit each other.
- In this process, the Generator learns to produce very good *fake data*.

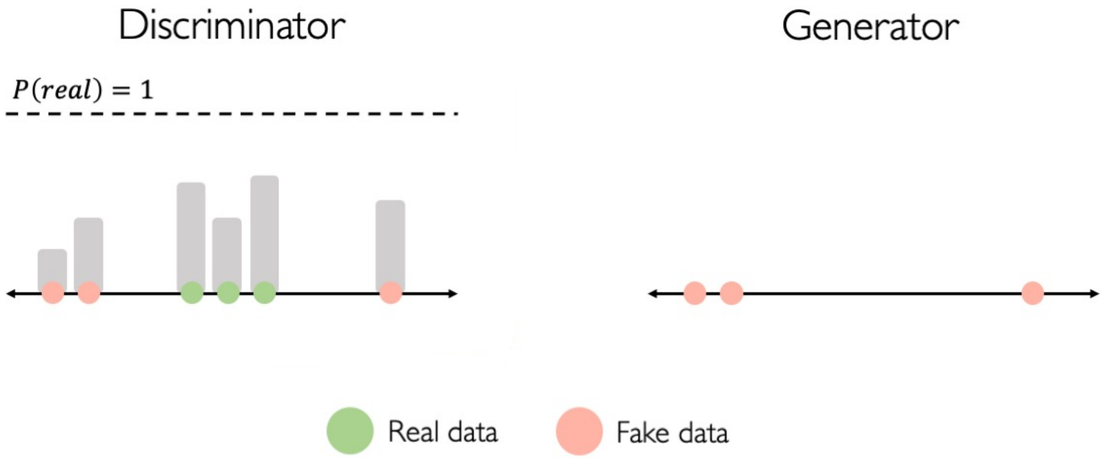
GAN: Learning



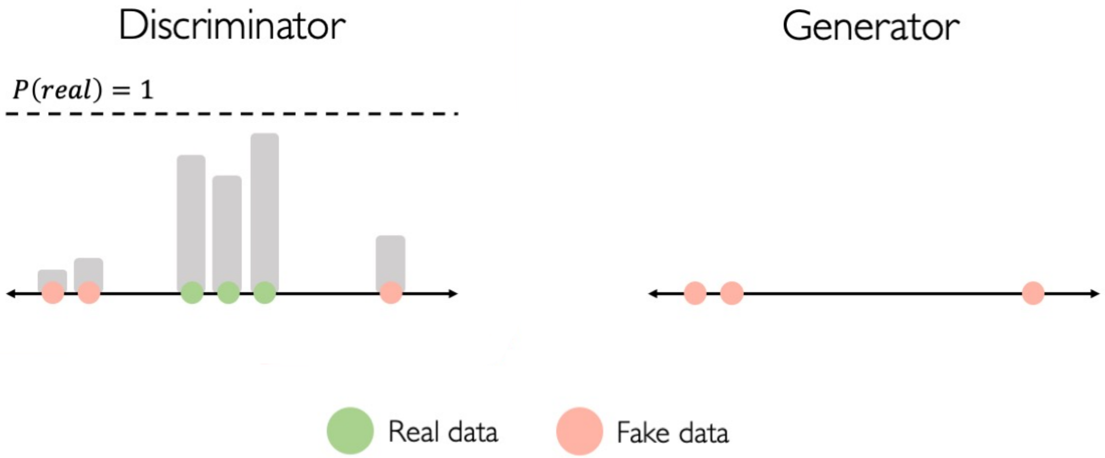
GAN: Learning



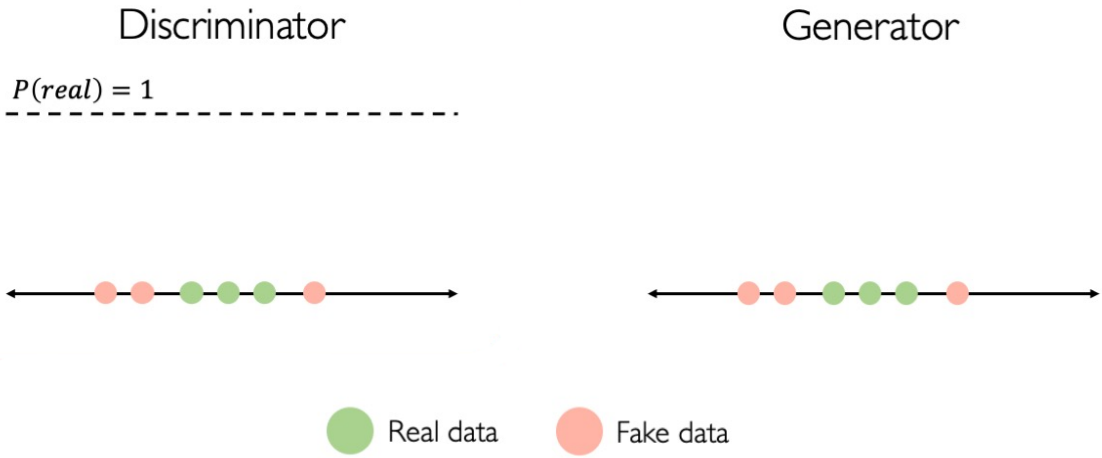
GAN: Learning



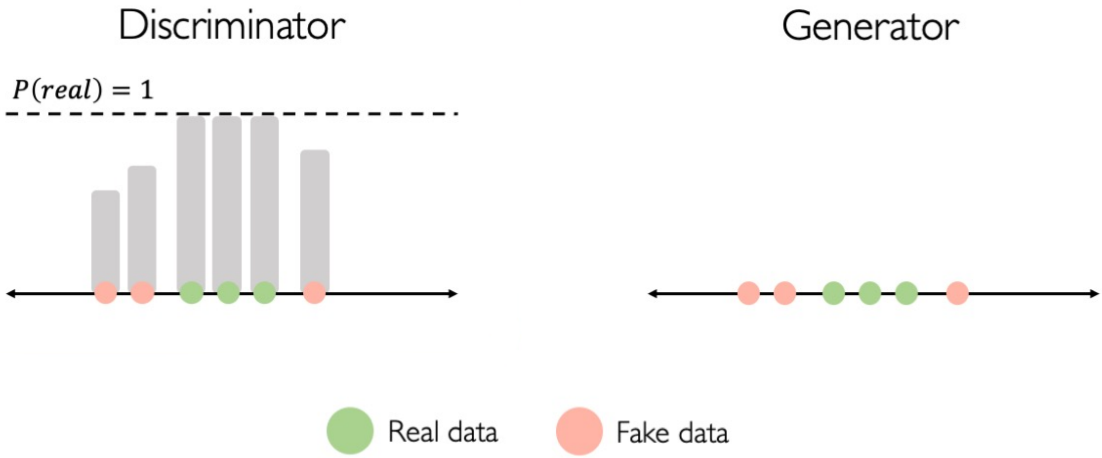
GAN: Learning



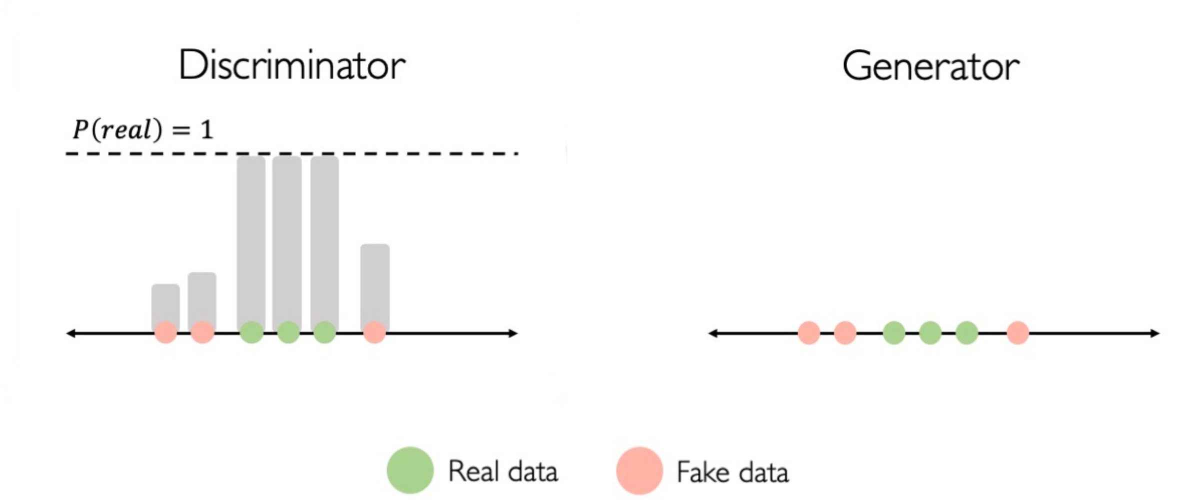
GAN: Learning



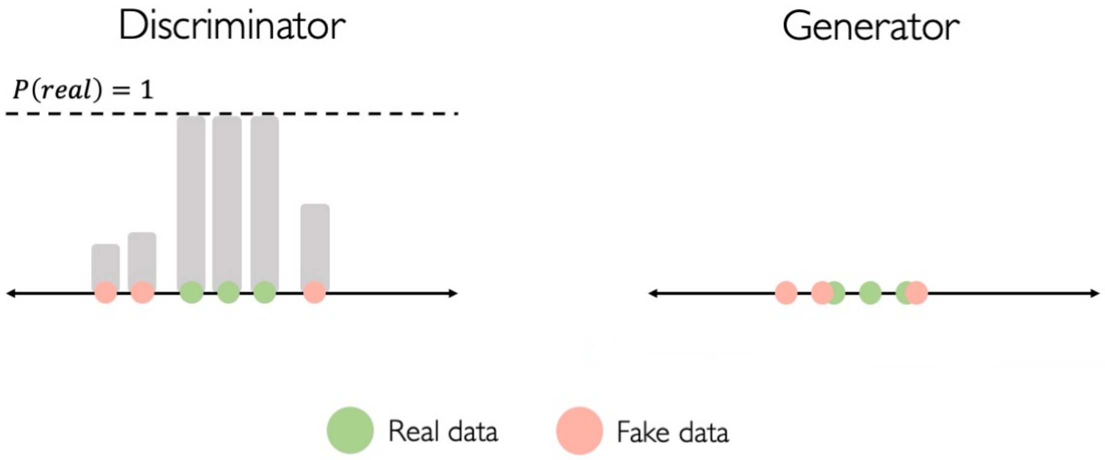
GAN: Learning



GAN: Learning



GAN: Learning



Generative Adversarial Network

- **Generator**: A Neural Network which given some *random noise* it generates a *fake datapoint*.
- **Discriminator**: A Neural Network that given an image determines if it is real or fake.

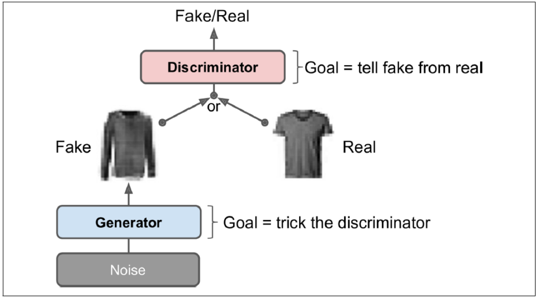


Figure 17-15. A generative adversarial network

Generative Adversarial Network: Loss Functions and Training

- A *Sampler* S samples a vector z in the *latent space* using a normal distribution.
- The Generator G maps z to a data point.

$$\hat{x} = G(z)$$

- The Discriminator D gets $\{\hat{x}, x\}$ where x is a training sample, i.e. a real datapoint.
- D outputs a probability for both $\hat{x}$ and x of them being real (i.e. not generated by G)

$$\hat{y} = D(\hat{x})$$

$$y = D(x)$$

- Note that $\hat{y}$ and y are probabilities

Generative Adversarial Network: Loss Functions and Training

- Discriminator's Loss:

$$\log(D(G(z))) + (1 - \log(D(x)))$$

- Generator's Loss:

$$1 - \log(D(G(z)))$$

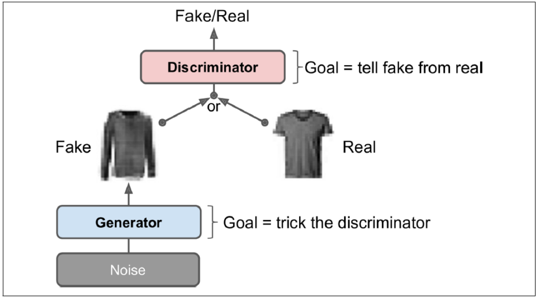


Figure 17-15. A generative adversarial network

Generative Adversarial Network: Loss Functions and Training

- Discriminator's Loss:

$$\log(D(G(z))) + (1 - \log(D(x)))$$

- Generator's Loss:

$$1 - \log(D(G(z)))$$

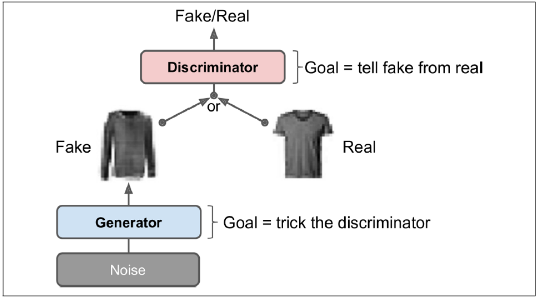


Figure 17-15. A generative adversarial network

- We train in rounds
- The Generator produces a batch of fake data $G(z)$
- A batch of real data x and $G(z)$ are both run through the discriminator, and loss $\log(D(G(z))) + (1 - \log(D(x)))$ is computed.
- We then update D via a gradient descent step, keeping G fixed.

Generative Adversarial Network: Loss Functions and Training

- Discriminator's Loss:

$$\log(D(G(z))) + (1 - \log(D(x)))$$

- Generator's Loss:

$$1 - \log(D(G(z)))$$

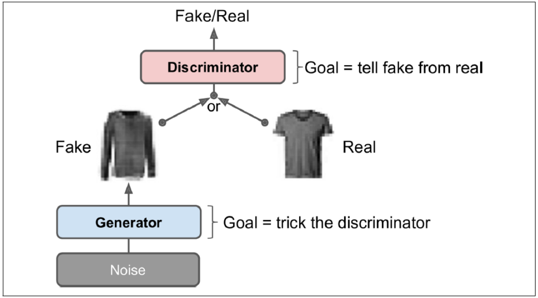


Figure 17-15. A generative adversarial network

- Next, the generator then produces a new batch of fake data $G(z')$
- These are processed by the discriminator to get the loss $\log(D(G(z')))$.
- We then update G by a gradient descent step, keeping D fixed.

Generative Adversarial Network: Loss Functions and Training

- Training GANs can be difficult, and we have to be very careful
- The training succeeds if we reach a good equilibrium between the Generator and the Discriminator.

- Training GANs can be difficult, and we have to be very careful
- The training succeeds if we reach a good equilibrium between the Generator and the Discriminator.

Some issues that may arise:

- *Mode Collapse:* The Generator only produces a small set of distinct images.
- *Convergence Failure:* The Generator produces poor quality images even after training for a long time. Could happen because the opponent network always counters whatever improvements you make, blocking every direction of progress in gradient descent
- *Losses don't indicate progress:* Even as the generator is improving, so is the discriminator. So the loss of the Generator may keep increasing even though it is getting better.

GANs for image synthesis: latest results

References:

These slides are based on:

- http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L4.pdf
- Generative Adversarial Networks, Goodfellow at al, Communications of ACM, November 2020