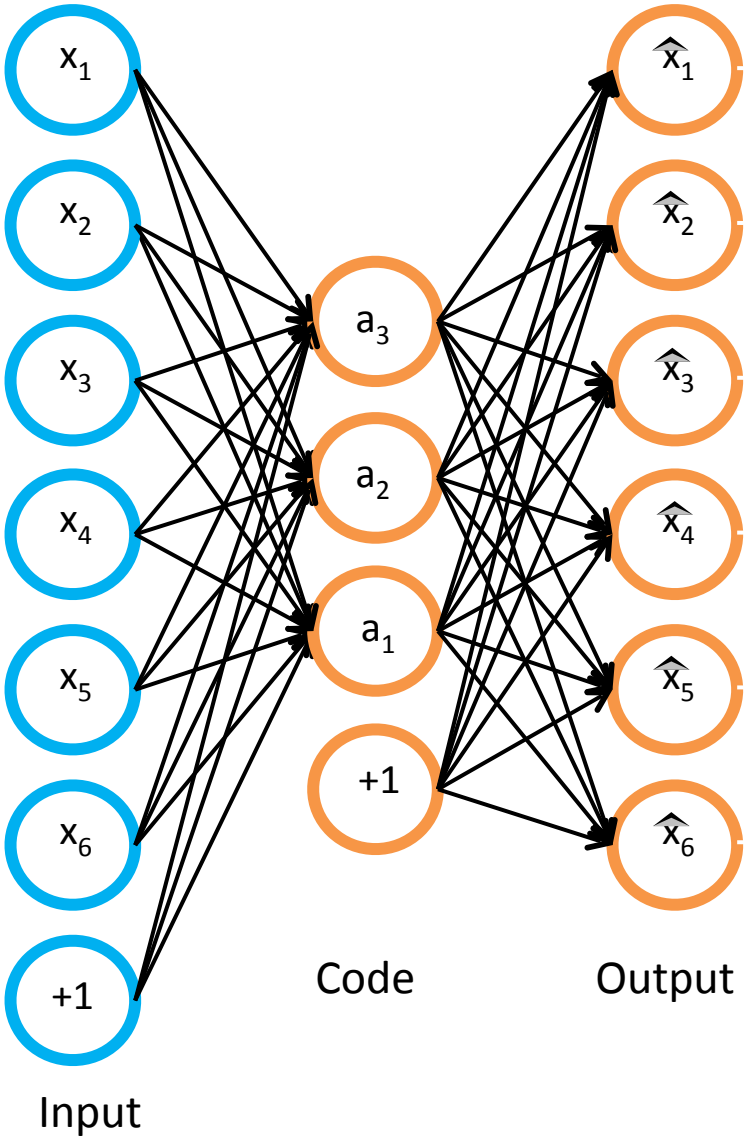
Lecture 17-18 Generative Adversarial Networks

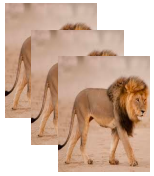
Ref: Outlier Analysis, Charu C Agrawal
Ref:Ian Goodfella Tutorial - https://arxiv.org/abs/1701.00160
Ref: Ian Goodfella Video Tutorial -https://www.youtube.com/watch?v=HGYYEUSm-0Q

Autoencoder Networks



Can we make a simple neural network for classification?

- E.g., design a network to discriminate humans and animals



Animals

Humans

x_1

x_2

x_3

...

...

x_{784}

Convert into a vector
28*28=784

FC

h_{11}

h_{12}

h_{13}

...

...

...

...

h_{1024}

Activation=ReLU
Dropout=0.2

FC

h_{21}

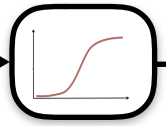
h_{22}

...

h_{256}

Activation=ReLU
Dropout=0.2

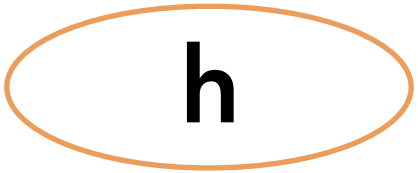
Σ



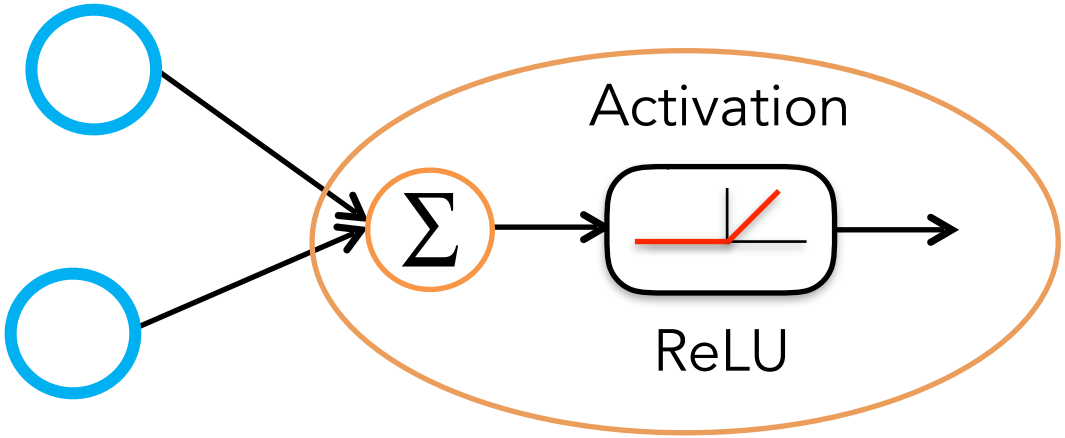
Sigmoid

Expected
1 for animals
0 for humans

Use back propagation for training!



- The hidden unit consists of summation unit and activation unit



- To avoid overfitting, we also specify dropout for each layer - force zero output from %p random units

Loss Function

Expected
1 for animals
0 for humans

- **Binary Cross Entropy - Let D() be the function represented by the network**

$$\mathscr{L}^d = \frac{1}{N}\sum_{i=1}^{N} y_i \; log \; \frac{1}{D(x_i)} + (1 - y_i) \; log \; \frac{1}{(1 - D(x_i))}$$

Penalty for mistake in detecting animals

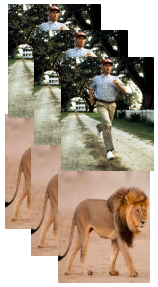
Penalty for mistake in detecting humans

Since we know the error (loss), we can back-propagate the error gradient and calculate the parameters!

Let us try to discriminate between real and fake images!

- Collect a large number of real images taken using cameras
- Collect a large number of fake images synthetically generated by computer
- Train the network

Discriminator - D()



Real

Fake

x_1
x_2
x_3
...
...
x_{784}

Convert into a vector
28*28=784

FC

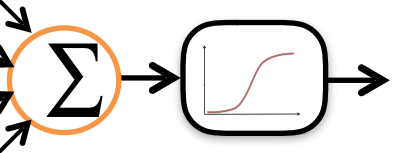
h_{11}
h_{12}
h_{13}
...
...
...
h_{1024}

Activation=ReLU
Dropout=0.2

FC

h_{21}
h_{22}
...
h_{256}

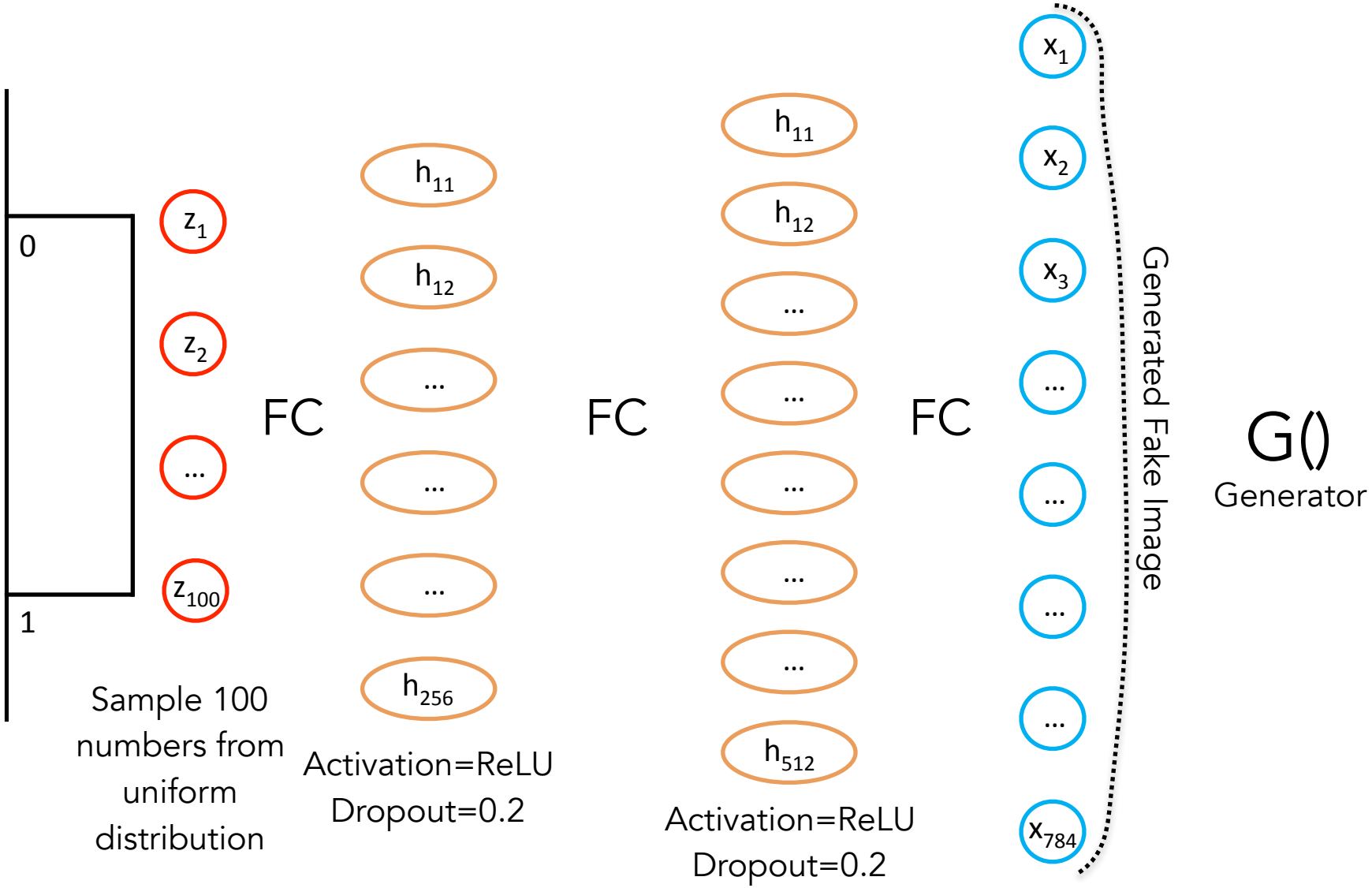
Activation=ReLU
Dropout=0.2



Sigmoid

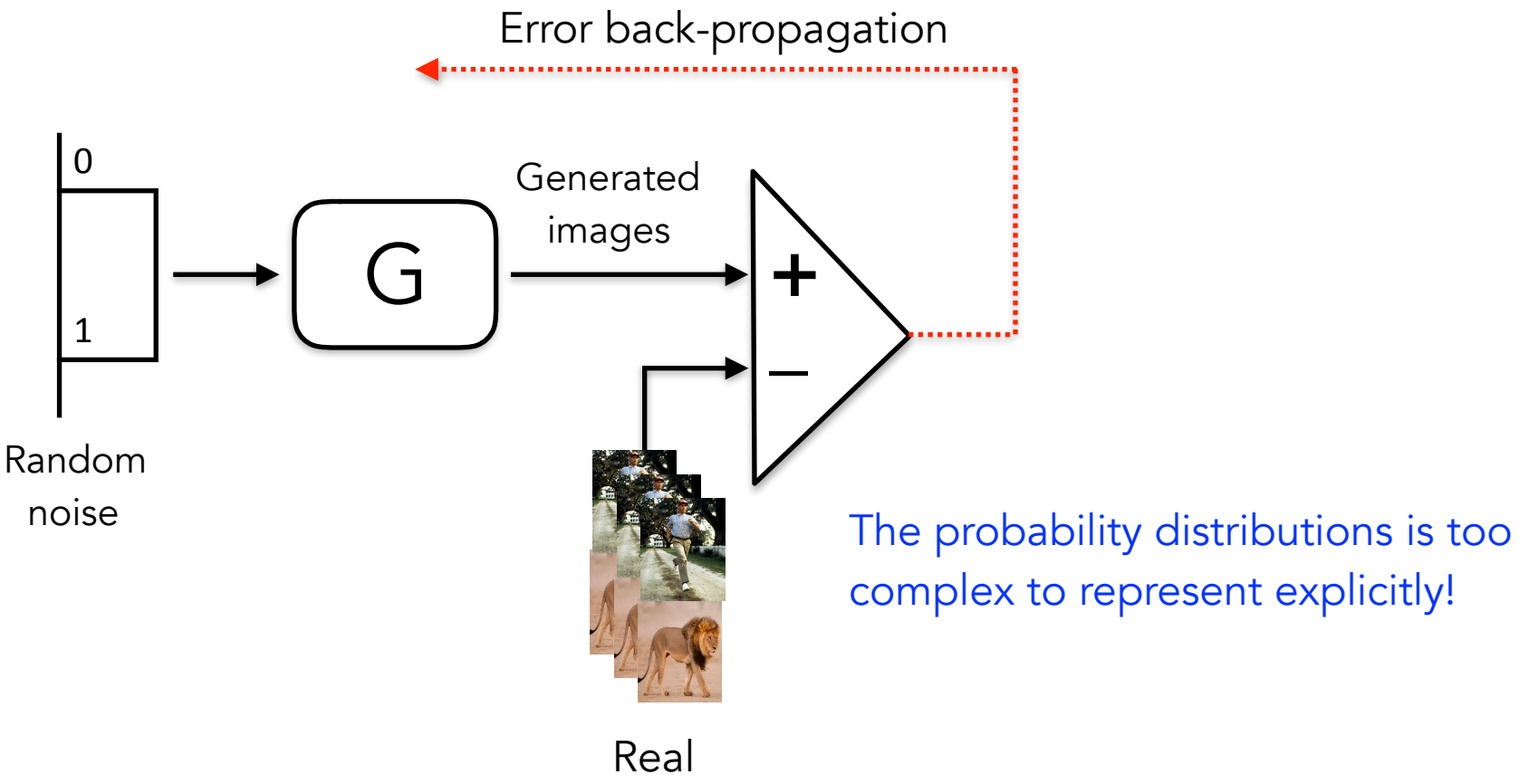
Expected
1 for real
0 for fake

Let us use the decoder part of the auto encoder to generate fake images!



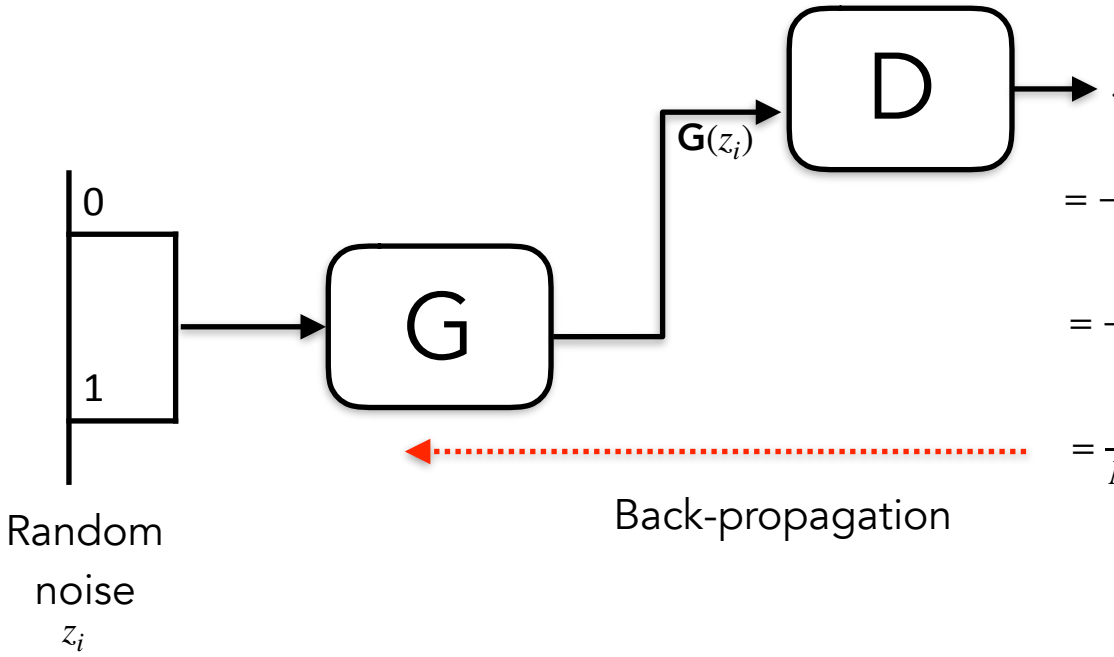
- If we use random weights, we will get a noisy fake image.
- Can we choose the weights such that the image is realistic?

Compare image distributions to calculate generator error



- We need an alternative approach to know how real the generated images is.
- Can we use the real/fake discriminator for this purpose?

Training Generator



Expected
1 for real
0 for fake

$$\mathscr{L}^g = -\mathscr{L}^d$$

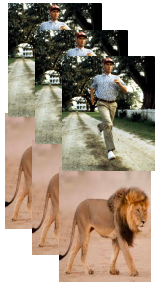
$$= -\frac{1}{N}\sum_{i=1}^{N} y_i \; log \; \frac{1}{D(G(z_i))} + (1 - y_i) \; log \; \frac{1}{(1 - D(G(z_i)))}$$

$$= -\frac{1}{N}\sum_{i=1}^{N} 0 \; log \; \frac{1}{D(G(z_i))} + (1 - 0) \; log \; \frac{1}{(1 - D(G(z_i)))}$$

$$= \frac{1}{N}\sum_{i=1}^{N} log \; (1 - D(G(z_i)))$$

- From where to get fake images to train discriminator?
- Let us use the images generated by generator as fake images to train discriminator!

Training Discriminator



Real

x_i

D

$G(z_i)$

G

0

1

Random noise z_i

Back-propagation

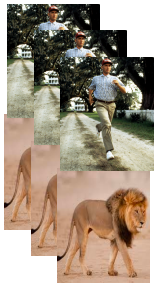
$$\mathscr{L}^d = \frac{1}{N}\sum_{i=1}^{N} log\ \frac{1}{D(x_i)} + log\ \frac{1}{(1 - D(G(z_i)))}$$

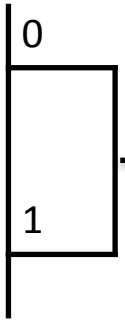
Expected
1 for real
0 for fake

Generative Adversarial Networks

1. Use trained discriminator to train generator
2. Use trained generator to train discriminator

Min-Max Game





G

x_i

$\mathbf{G}(z_i)$

D

$$\mathscr{L}^d = \frac{1}{N}\sum_{i=1}^{N} log\ \frac{1}{D(x_i)} + log\ \frac{1}{(1 - D(G(z_i)))}$$

$$\mathscr{L}^g = \frac{1}{N}\sum_{i=1}^{N} log\ (1 - D(G(z_i)))$$

When to stop?

- When you reach Nash Equilibrium, I.E., D() START producing 0.5 for all samples
- IN practice, visually inspect the generated images how realistic they are

Question: What does the generator network represent after training?

Inverse CDF?

How detect Anomaly using GANs?

- Train GAN by using normal data
- Take the test image and map it to the latent space (z)
- The difference of generated image and original image can be anomaly score
- The discriminator loss is also in indicator of anomaly

How to obtain latent space representation?

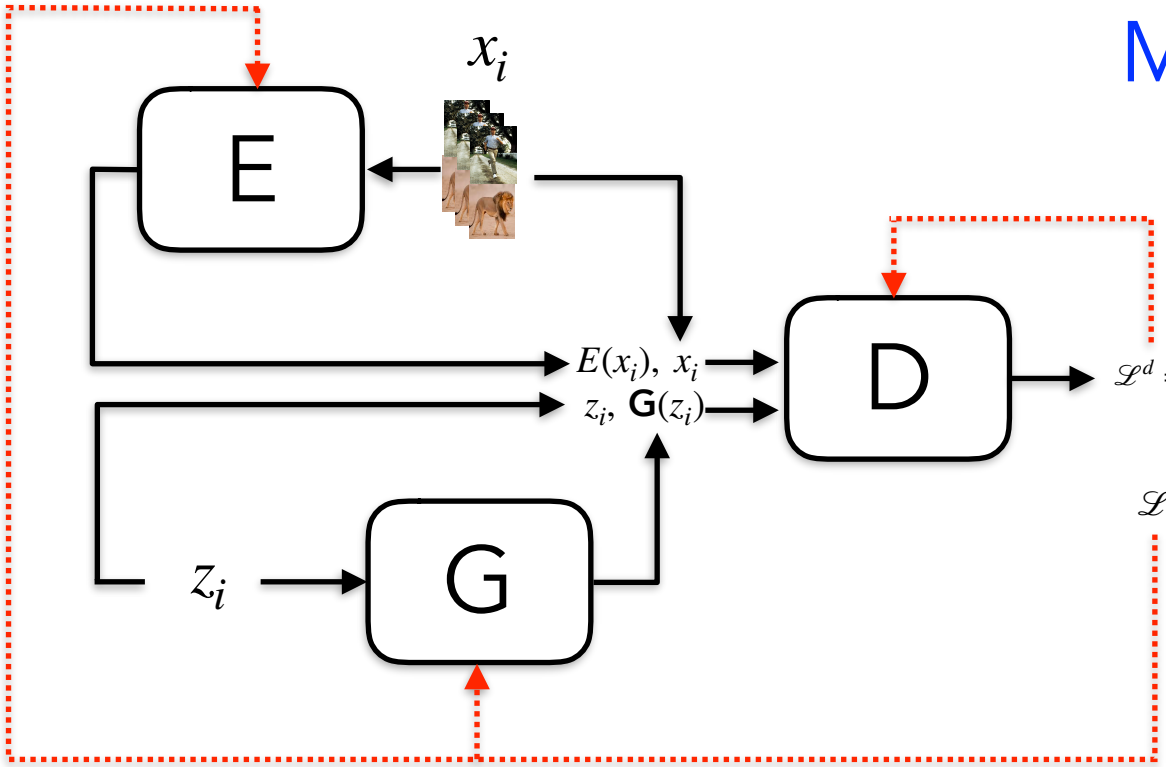
- Generate a random vector z
- Use generator to generate image G(z)
- Measure residual loss = |G(z)-x|
- Feed G(z) to discriminator and measure the generator loss $\mathscr{L}^g$
- Use weighted sum of both the losses to

AnoGAN

- Generate a random vector z
- Use generator to generate image G(z)
- Measure residual loss = |G(z)-x|
- Feed x and G(z) to discriminator
- Discriminator loss = |f(x)-f(G(z))|, f is intermediate layer
- Use sum of both the losses to

BiGAN

1. E and G minimise $-\mathscr{L}^d$
2. D minimizes $\mathscr{L}^d$



Min-Max Game

$$\mathscr{L}^d = \frac{1}{N}\sum_{i=1}^{N} log\ \frac{1}{D(E(x_i), x_i))} + log\ \frac{1}{(1 - D(z_i, G(z_i)))}$$

$$\mathscr{L}^g = \frac{1}{N}\sum_{i=1}^{N} log\ (1 - D(z_i, G(z_i)))$$